Conception interactive d'environnements urbains durables à base de résolution de contraintes

Bruno Belin

l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Bruno BELIN

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et technologies de l'information, et mathématiques

Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire d'informatique de Nantes-Atlantique (LINA)

Soutenue le 27 novembre 2014

Conception interactive d'environnements urbains durables à base de résolution de contraintes

JURY

Rapporteurs :	**M. Gilles TROMBETTONI**, Professeur des Universités, Université de Montpellier 2
	M. Emmanuel PRADOS, Chargé de Recherche INRIA, HdR, INRIA Grenoble - Rhône-Alpes
Examinateurs :	**M. Frédéric SAUBION**, Professeur des Universités, Université d'Angers
	M. Etienne GAUDIN, Directeur Innovation, Groupe Bouygues
	M. Marc CHRISTIE, Maître de conférences (Co-encadrant), Université de Rennes 1
	M^me^ Charlotte TRUCHET, Maître de conférences (Co-encadrante), Université de Nantes
Directeur de thèse :	**M. Frédéric BENHAMOU**, Professeur des Universités, Université de Nantes

Remerciements

Recherche un jour, recherche toujours : commençons par remercier mes initiateurs à la Recherche, en somme les deux responsables de toute cette histoire :

- Christophe JERMANN, Colin DE LA HIGUERA.

Toujours là pour assurer le combat, prenons maintenant la direction du LINA :

- Pierre COINTE, Anne-Françoise QUIN, Annie BOILOT, Jean-Paul SACHET ;
- Élodie GUIDON, Floriane PIQUET, Christine BRUNET.

Lorsque mon PC a bobo, ou que les imprimantes restent muettes, on les retrouve jusqu'à la cafette :

- Damien VINTACHE, Jean-Yves LEBLIN, S@bine BEAURAIN.

Tous au restaurant U, c'est le repère des doctorants U :

- mes collègues doctorants actuels qui devraient bientôt me demander des conseils ;
- les anciens qui m'ont donné des conseils d'anciens ;
- et l'unique, la formidable, l'inoubliable Marie PELLEAU bien-sûr.

Pause, là c'est l'heure du bon café, des conseils et recommandations en tout genre, des présentations scientifiques, des choses qui ne s'inventent pas tout seul :

- tous mes collègues de l'équipe TASC ;
- Éric LANGUENOU, Florian RICHOUX et Benoît DELAHAYE.

Évoquons maintenant mes partenaires durables, ceux impliqués avec moi dans le projet de recherche SUSTAINS :

- Élisabeth POINTAL et Frédéric VERNIER du LIMSI : Google Map, et c'est par là que tout commença ;
- Julian JOSEPH de l'Université Rennes 1 : SOS C++, vous êtes toujours là ?
- Valérie GACOGNE, Bernard COUTROT et Michel FANNI : trois mousquetaires des villes modernes ;
- Erwan MAHÉ, Romain CAVAGNA, David KNOSSOW : les gars d'Artefacto qui font les plans les plus beaux (villes en 3D).

Et pour finir, mes Boss de ces trois dernières années qui m'ont fait bosser, bosser et bosser : mais ça c'est normal pour un Boss, non ? Ils ont bien mérité une ligne dédiée :

- Frédéric BENHAMOU ;
- Charlotte TRUCHET ;
- Marc CHRISTIE.

Je remercie également mes rapporteurs qui ont accepté d'évaluer ce manuscrit et l'ensemble de mon jury qui me fait l'honneur de s'intéresser aux travaux de cette thèse.

Et pour tous ceux que j'aurai oublié de citer, qui que vous soyez, un grand merci !

Un tout dernier mot pour me faire pardonner de mes absences répétées auprès de mes amis, de ma famille, de ma femme et mes trois filles adorées.

Table des matières

1

Introduction

1.1 Contexte général

Depuis peu, les villes rassemblent la majorité de la population mondiale et ne cessent de se développer, en particulier dans les pays émergents. En même temps qu'elles absorbent les populations venant des campagnes, elles concentrent des difficultés qui peuvent être liées au prix des logements, aux embouteillages, à l'insuffisance ou à la vétusté des équipements publics, au niveau de bruit, à la pollution de l'air ou encore à l'insécurité, etc. Parallèlement, les villes ont subi des transformations importantes liées aux nouveaux modes de transport qui ont permis de repousser les distances pouvant être parcourues quotidiennement. Ceci se traduit par des citadins, de plus en plus nombreux, contraints de s'éloigner un peu plus loin, repoussant en même temps les limites des villes qui s'étalent sans fin. Mais les nouveaux moyens de transport ne sont pas les seuls responsables de la métamorphose des zones urbaines, les politiques successives de la ville et les nouveaux modes de vie y contribuent également pour une large part. Par exemple, le zoning strict insufflé par « La Charte d'Athènes » [55], une doctrine des années 40 élaborée par LE CORBUSIER[1] qui prescrit un classement des grandes fonctions urbaines par catégories, a connu ses heures de gloire mais nous a légué un héritage qui est souvent encombrant, laissant pour compte des cités entières coupées du reste de la ville par des voies de communication devenues infranchissables.

Indépendamment du bien être de ces habitants, nos modes de vie actuels laissent apparaître d'autres défis à l'échelle de la planète entière qui viendront s'ajouter aux difficultés auxquelles les villes sont déjà confrontées. Trop souvent détourné pour des intérêts opportunistes ou repoussé à plus tard, la mise en œuvre du développement durable tarde à s'encrer dans nos sociétés. Ses partisans nous mettent pourtant en garde face aux effets prévisibles du réchauffement climatique, de la montée du niveau des océans ou encore de l'appauvrissement de la biodiversité ou des ressources non renouvelables, etc. Les conséquences annoncées ont longtemps été contestées, la preuve scientifique n'ayant pas été apportée. Néanmoins, constatons ensemble que ces voix se font de plus en plus discrètes ces dernières années. Toujours plus éloigné de la campagne, l'Homme croit, à tort, pouvoir dominer la Nature et s'en abstraire. Il l'exploite sans trop se soucier des dégâts irréversibles qu'il cause en reportant sur les générations futures un fardeau qui impactera durablement leurs conditions de (sur)vies.

La majorité des activités humaines s'opérant désormais dans les villes, leur adaptation représente un levier extraordinaire pour limiter ou lutter contre les effets à venir. Dans ce domaine, deux axes de travail

[1] Architecte urbaniste, de son vrai nom Charles-Édouart Jeanneret-Gris, né en 1887. Il s'intéresse notamment à l'industrialisation des bâtiments collectifs et à la fabrication standardisée d'équipements en série. Ses travaux prendront valeur de solution après la seconde guerre mondiale en France pour répondre à la crise du logement et reconstruire dans l'urgence.

complémentaires sont envisageables : soit on intervient à l'occasion du renouvellement urbain, soit on intègre les concepts d'un développement durable dans les fondations des villes naissantes. Le renouvellement urbain s'opère sur le long terme. Néanmoins, des expérimentations sont déjà menées à l'échelle de quartiers labellisés « écoquartier ». Ce mouvement en plein développement est né en Allemagne et en Hollande avec quelques exemples pionniers en Autriche, en Grande Bretagne et dans les pays Scandinaves, et il a fini avec quelques décennies de retard par atteindre la France. Et bien que la création de villes nouvelles ne soit plus de mise dans nos pays déjà fortement urbanisés, on constate néanmoins dans des pays à fort développement, que des villes importantes peuvent apparaître en l'espace de quelques années seulement, ce phénomène allant perdurer sur plusieurs décennies. C'est le cas notamment de la Chine qui prévoit de créer l'équivalent de vingt nouvelles villes d'un million d'habitants chacune et par an jusqu'en 2020 pour y accueillir ses paysans qui désertent les campagnes [182].

La modélisation urbaine et l'optimisation combinatoire peuvent accompagner les décisions stratégiques qui engagent, chacune à son niveau, notre devenir collectif. La conception de villes nouvelles[2] est par nature une tâche collaborative qui rassemble des urbanistes, des aménageurs et des décideurs autour d'une carte représentant un territoire à aménager. Cette tâche consiste à positionner sur la carte des éléments urbains correspondant aux centres, habitations, commerces, équipements publics, aux zones industrielles ou commerciales, etc. Mais pour cela, le nombre d'éléments urbains à positionner sur le territoire, tout comme leur répartition spatiale, nécessitent d'établir des règles complexes issues d'une vision systémique de la ville pour y intégrer les contraintes relatives à un développement durable avec une prise en compte simultanée de ses aspects sociaux, économiques et écologiques. Enfin, ces travaux doivent s'inscrire dans une réflexion interdisciplinaire pour se doter des compétences et des savoirs croisés en rapport avec la complexité de nos sociétés modernes.

1.2 Objectifs et contributions de cette thèse

Face à la complexité intrinsèque liée aux agglomérations urbaines, nous cherchons à apporter des réponses concrètes circonscrites à des questions précises. Nous nous focalisons plus particulièrement sur la problématique de placement des éléments urbains à l'aide de techniques interactives et d'optimisation combinatoire.

1.2.1 Objectifs

Cette thèse s'inscrit dans un projet de recherche national qui rassemble des urbanistes, des architectes, des experts du secteur de l'énergie et des équipes autour de l'optimisation combinatoire et de l'interaction homme-machine. Nous voulons donc exploiter au mieux cette interdisciplinarité pour mener à bien un travail collaboratif profitable à chacun, et qui puisse être utile.

Notre travail consiste à réaliser un système interactif d'aide à la décision capable de répartir au mieux les différentes entités composant une programmation urbaine (habitat, industrie, équipement, commerces, etc.) sur un territoire vierge et délimité dans l'espace. Cet outil s'adresse aux urbanistes, aménageurs urbains et décideurs. Quelle que soit la catégorie d'utilisateur, le logiciel proposé doit être capable de l'assister dans cette tâche de conception de villes durables tout en l'informant des conséquences de ses choix.

1.2.2 Contributions

Nos contributions portent sur la modélisation du problème d'aménagement urbain, la résolution initiale de ce problème y compris pour des instances de grande taille et la proposition d'un système permettant de maintenir la cohérence des solutions dans les étapes interactives.

[2] Bien qu'il ne s'agisse pas de la même échelle, nous envisageons la création d'un nouveau quartier d'une façon similaire à la création d'une nouvelle ville, en tenant compte en plus de l'influence que peuvent exercer les quartiers limitrophes existants sur la zone à aménager.

Modélisation Dans un premier temps, nous analysons avec les urbanistes les principales règles urbaines et prenant en considération les supposés implicites passés sous silence de prime abord. Nous exprimons ensuite ces règles urbaines sous la forme de contraintes spatiales, chaque contrainte étant associée à une « fonction de coût » capable d'évaluer le niveau de violation des propriétés urbaines. Une description formelle de chaque contrainte est proposée. Un travail important consiste à exprimer chaque contrainte en vue d'une résolution efficace.

Résolution initiale Dans un deuxième temps, notre problème d'aménagement urbain étant modélisé comme un problème d'optimisation, nous le résolvons par des techniques de recherche locale basées sur la méthode *Adaptive Search*. Il s'agit d'une méthode générique basée contraintes (Constraint-Based Local Search) capable de tirer parti de la structure du problème qui lui est confié pour diriger la recherche vers une solution optimale ou de bonne qualité en un temps raisonnable. Nous proposons également des heuristiques capables d'améliorer sensiblement les temps de calcul. Pour passer à l'échelle et travailler avec des villes réelles, différents protocoles sont élaborés pour décomposer le problème global et répartir les traitements d'optimisation sur une grille de calcul.

Tous ces travaux permettent d'obtenir de façon automatique différentes propositions (solutions initiales optimisées) pour un même problème.

Interaction avec les utilisateurs Notre système est destiné à être utilisé dans un contexte d'aide à la décision. Il doit assister les urbanistes, aménageurs et décideurs à établir des plans de villes durables, sans pour autant se substituer à eux. Il est donc fondamental qu'ils puissent interagir avec la solution en utilisant un dispositif tactile adapté. A ce stade, l'opérateur sélectionne une solution parmi différentes propositions pour l'ajuster à ses besoins et pour qu'elle puisse s'adapter automatiquement aux arbitrages opérés entre les différents acteurs impliqués dans le projet urbain. À chaque fois qu'un opérateur réalise une modification sur la ville, le système recalcule à la volée une bonne solution proche de la précédente sans remettre en cause les choix antérieurs imposés par les utilisateurs. En retour des modifications réalisées par les utilisateurs, des graphiques leurs sont également fournis (carte de température, évolution des coûts par contrainte) pour représenter les impacts de leurs choix.

En lien avec ces différents travaux, nous avons également chercher à évaluer nos résultats suivant trois aspects complémentaires :

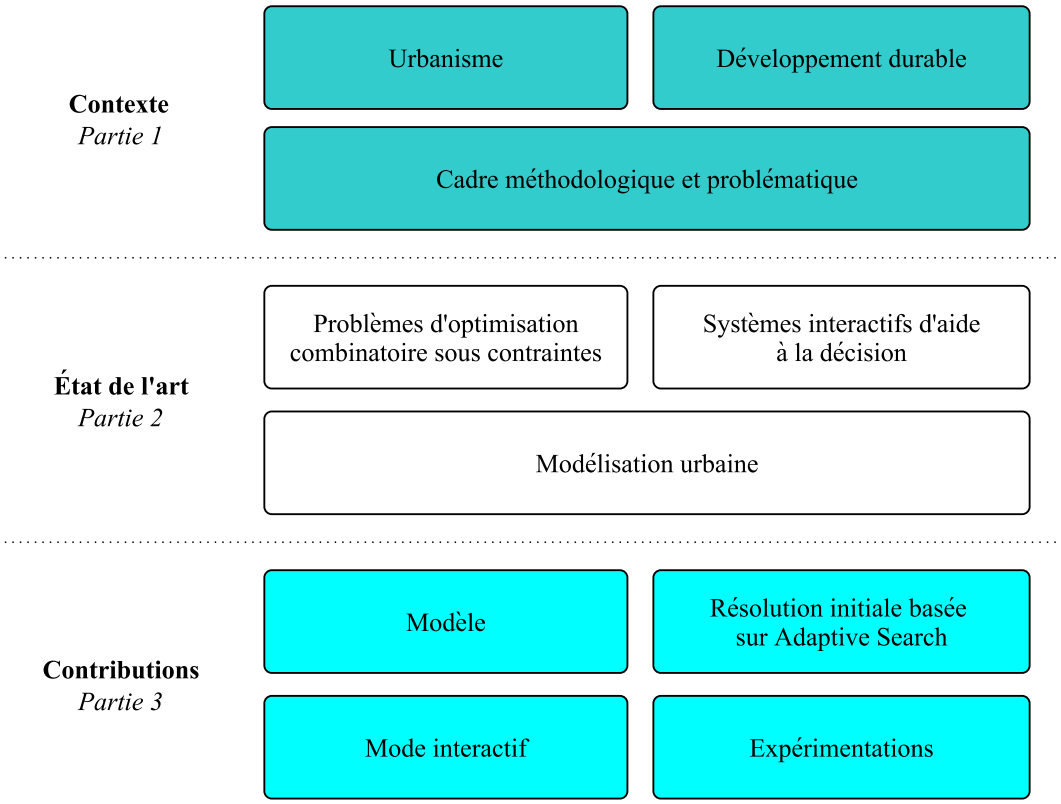
- les temps pour une résolution initiale en comparant le mode séquentiel au mode distribué ;
- la qualité des solutions produites automatiquement par la résolution initiale ;
- la pertinence de l'interaction et la réactivité globale du système.

Le premier aspect est réalisé par une série d'expérimentations alors que les deux aspects suivants s'appuient sur les retours des urbanistes impliqués dans le projet de recherche SUSTAINS.

1.3 Plan de la thèse

Notre présentons dans cette section le plan de cette thèse organisée en trois parties. La première partie reprend les notions essentielles liées à l'urbanisme et au développement durable et se termine par une présentation de notre cadre méthodologique et de la problématique qui nous intéresse. La seconde partie dresse un état de l'art complet des problèmes d'optimisation combinatoire sous contraintes, des systèmes interactifs d'aide à la décision et de la modélisation urbaine. La troisième partie présente notre travail de modélisation, les techniques mises au point pour une résolution initiale basée sur Adaptive Search, le mode interactif et les différentes expérimentations réalisées. Tous ces éléments sont repris et illustrés dans la figure 1.1.

FIGURE 1.1 – Organisation de la thèse.



Partie I : Contexte Nous commençons par faire un inventaire des notions de base relatives à l'urbanisme en revisitant les transformations profondes et successives de nos sociétés. Nous définissons notamment ce qu'est une ville et un espace urbain avant d'aborder les différents courants qui ont traversés l'urbanisme et qui ont participés à la transformation des villes jusqu'à aujourd'hui.

La notion de développement durable n'existe que depuis une vingtaine d'années et malgré cette jeunesse, elle a été utilisée dans bien des circonstances et on ne sait pas toujours très bien ce que revêt cette expression. Il nous semble donc important de revenir sur les enjeux d'un développement soutenable sur le long terme pour se focaliser ensuite sur les dimensions urbaines du développement durable.

A ce stade, nous somme en mesure d'évaluer avec plus de précision ce que l'urbanisme peut apporter au développement durable. Nous présentons également les grandes lignes du projet de recherche SUSTAINS qui s'inscrit dans la conception de villes nouvelles durables à un stade très en amont du processus de planification. Compte tenu de ces précisions, nous pouvons i) fixer un cadre méthodologique qui s'articule autour d'une plateforme globale d'aide à la décision pour la conception d'environnements urbains durables et ii) dégager une problématique précise liée à cette thèse.

Partie II : État de l'art Notre problème étant de nature combinatoire, nous identifions les propriétés relatives à ce type de problème et dressons un inventaire des principales méthodes de résolution capables de les adresser. Nous insistons plus particulièrement sur les métaheuristiques de recherche locale particulièrement adaptées à notre contexte.

Pour aider les utilisateurs à prendre des décisions, nous décryptons les systèmes interactifs d'aide à la décision qui permettent de concevoir des systèmes temps réel capables de fournir des résultats dans un délai contrôlé. On y évalue également l'intérêt d'une approche de résolution *anytime* à même de prendre en compte nos exigences liées à l'interactivité.

Enfin, on présente un état de l'art de la modélisation urbaine restreint à des sujets en lien avec notre

thèse. Trois approches (*UrbanSim*, *GeOpenSim*, et *CommunityViz*) sont détaillées afin de positionner nos travaux dans le panorama de la recherche actuelle. En complément de ces modèles urbains, nous présentons également la famille des problèmes de positionnement et notamment le problème d'affectation quadratique qui offre des similitudes avec notre problématique, qui consiste à positionner au mieux les éléments urbains sur chaque îlot d'une ville.

Partie III : Contributions Nous commençons par définir les différentes notions retenues pour représenter une pré-programmation urbaine, et en particulier : l'îlot, la centralité, la forme urbaine et l'intensité. En partant de ces notions, nous proposons un modèle de contraintes pour notre problème d'aménagement urbain. Cette étude fournit une description informelle associée à une description formelle pour chaque contrainte, ces contraintes pouvant être regroupées selon qu'elles sont générales ou spécifiques.

Le problème d'aménagement urbain étant modéliser comme un problème d'optimisation, nous le résolvons par des techniques de recherche locale en réalisant une succession de permutations entre formes urbaines. L'outil proposé organise alors automatiquement les formes urbaines sur un territoire vierge et délimité dans l'espace. L'algorithme séquentiel basé sur *Adaptive Search* bénéficie d'heuristiques novatrices permettant d'améliorer les temps de calcul. Nous proposons également une version distribuée de l'algorithme efficace pour s'attaquer à des problèmes de grande taille.

La seconde partie du processus de résolution est interactive. Après la génération d'une solution initiale par des procédures complètement automatiques, l'utilisateur peut reprendre le contrôle sur la solution par le biais de manipulations interactives qui sont décrites ici dans les moindres détails. Les trois composants impliqués dans l'interaction sont également présentés : dispositif tactile, solveur interactif et gestionnaire de messages asynchrones.

Après avoir décrit notre méthodologie d'évaluation, le dernier chapitre fait état des expérimentations liées aux résolutions initiales. Dans ce cadre, on y reporte également les performances du mode distribué, le seul mode capable de s'attaquer efficacement à des villes de grande taille. Nous nous focalisons ensuite sur le retour des urbanistes qui ont pu évaluer la qualité des différentes solutions initiales produites sur Marne-La-Vallée. Nous fournissons également leurs réponses liées à la mise en œuvre du mode interactif qui a été particulièrement bien accueilli.

Contexte

2

Urbanisme

Nous débutons cette partie liée au contexte en faisant l'inventaire des notions de base relatives à l'urbanisme [69, 261] qui sont au cœur de ce travail de thèse.

Nous avons connu un premier bouleversement en 1995 lorsque les villes des pays industrialisés qui représentaient jusque là les plus grosses concentrations humaines ont été rattrapées par celles des pays émergents [41]. Une autre étape importante a été franchie en 2008 lorsqu'on a atteint pour la première fois dans l'histoire de l'humanité le cap des 50 % de la population mondiale vivant en agglomération dont un tiers en bidonville. En 2010, notre planète comptait 449 agglomérations de plus d'un million d'habitants, dont 38 entre 5 et 10 millions et 23 avec plus de 10 millions d'habitants [270].

Aujourd'hui, les villes des pays émergents totalisent 95 % de la croissance urbaine mondiale et on dénombre un million d'urbains supplémentaires chaque semaine dans le monde. L'Amérique et l'Europe sont les plus urbanisées, tandis que l'Afrique et l'Asie comptent encore une majorité de ruraux. L'Europe de l'Ouest est globalement très urbanisée avec un taux de 80 %. En France, le taux d'urbanisation actuel est de 87 % [269]. Nous sommes dans un milieu très urbain, mais dans une urbanité de moins en moins dense et qui intègre un usage urbain de la campagne.

Dans les projections publiées par les NATIONS UNIES [271], la population mondiale en zone urbaine devrait atteindre 58 % en 2025, puis 67,2 % en 2050. Cette augmentation sera principalement concentrée dans les régions les moins développées : Afrique et Asie. Cette croissance ne sera pas sans conséquences sur les villes qui devront continuer à s'adapter pour assumer leurs responsabilités dans des domaines aussi variés et indissociables que sont le logement, la sécurité, les infrastructures de transport et de télécommunication, l'approvisionnement en eau et en énergie, la qualité de l'air, le niveau de bruit ou encore l'évacuation et le traitement des déchets [141].

2.1 Qu'est-ce qu'une ville ?

Définitions JEAN-LOUIS HUOT, professeur d'archéologie, esquisse une définition de la ville [140] : c'est un lieu d'échanges et de production, un lieu d'interactions économiques complexes. C'est un lieu mixte et hiérarchisé où plusieurs classes sociales coexistent et la société y est organisée de manière structurée. Selon lui, une vraie ville doit être mixte, complexe et hiérarchisée. Pour les statisticiens français, depuis 1887, les choses sont simples : une ville est une agglomération comptant au moins 2 000 habitants. C'est la même chose en Allemagne, mais il existe près de 200 seuils à travers le monde. Pour l'historien Belge HENRI PIRENNE, les villes sont l'œuvre des marchands, elles n'existent que par eux [217]. Pour l'historien français PIERRE LAVEDAN, la ville est un être vivant [166]. Comme tout être vivant, elle naît, elle grandit,

elle meurt. PIERRE RIBOULET, architecte et urbaniste français, parle d'œuvre [232] : la ville est une œuvre collective, une sorte de production artisanale unique, réalisée par les hommes qui composent la société et qui y vivent.

Principes constitutifs de la ville PIERRE MERLIN, ingénieur géographe, expert-démographe et statisticien, écrit [188] que la ville est constituée de deux éléments : l'espace et le temps. A ces deux composantes, il faut ajouter celle des dynamiques socio-économiques. La dynamique sociale et économique, c'est la société en mouvement dans l'espace et dans le temps. Ce sont les initiatives et les mouvements de la société, des groupes sociaux et des individus qui la constituent, qui produisent la ville et la font évoluer à la fois dans le temps et dans l'espace. Ces dynamiques intègrent une quadruple dimension : économique, culturelle, démographique et juridique. Le temps est l'élément qui fait évoluer la ville dans la durée, qui fonde progressivement sa complexité. L'espace quant à lui est le support matériel de la ville : le site, l'environnement naturel et les formes d'occupation du sol. Les formes urbaines partagent quelques caractéristiques de base :

- le tracé des voies et des espaces publics, support des flux et des réseaux ;
- le découpage parcellaire, support matériel et juridique de l'investissement social du territoire ;
- l'occupation du sol, c'est-à-dire les caractéristiques du bâti et des aménagements des surfaces non bâties.

L'évolution urbaine "naturelle" est celle du "marché", celle qui traduit les rapports de forces au sein de la société. Mais cette évolution peut être corrigée par les nécessités de l'intérêt général, et l'urbanisme avec la politique sont des instruments de cette correction. Pour la sociologue JANE JACOBS, le dynamisme économique des villes se fait à cause de ses congestions et de ses dysfonctionnements [142], car c'est la richesse du creuset urbain, son bouillonnement qui conduisent à la création économique et culturelle : plus la ville est grande, plus les fonctions urbaines sont diversifiées, plus elle a de chance d'inventer de nouveaux métiers, de nouvelles productions, donc de continuer à croître.

Le droit qui régit à la fois la possession et l'usage du sol est un élément majeur qui façonne la ville de manière essentielle, à la fois dans l'espace et dans le temps. C'est donc un élément fondateur de la ville.

L'homme est un être social complexe qui, au-delà de ses besoins de base, a des besoins spécifiques qui varient d'une culture à une autre, d'une époque à une autre, d'un groupe social à un autre. Ce sont les sociétés organisées qui peuplent la ville et la font évoluer en fonction de leurs besoins, de leurs désirs et de leurs ambitions. Comme le rappelle JACQUES DONZELOT [77], les ghettos sont d'abord issus du choix de « l'entre soi » des plus riches. Ainsi, les évolutions sociologiques les plus récentes en France, comme l'augmentation du nombre de divorces, de familles monoparentales, la diminution du nombre de personnes par foyer et par logements, et l'augmentation globale de la richesse et des exigences en matière de confort au cours des quarante dernières années, ont conduit à une modification radicale des besoins en terme de logements, de services urbains, de transport.

Finalement, la ville est intimement liée à l'évolution démographique parce que c'est elle qui forge la croissance ou le déclin de la population qui est un facteur qui interagit fortement avec les dynamiques économiques et sociales. Lorsqu'une dynamique de croissance économique existe, la croissance de la population vient souvent la renforcer. La ville n'est pas un support statique pour l'économie et la population. Elle est en perpétuelle évolution et il existe une interaction forte entre les différentes composantes économiques, démographiques et sociales.

2.2 La ville perçue comme une entité vivante

Les villes naissent et se développent. Elles peuvent aussi avoir des périodes de décroissance ou de stagnation et, enfin, elles peuvent mourir. Après une catastrophe, les habitants les abandonnent pour s'installer ailleurs, à côté ou beaucoup plus loin.

Naissance des villes Une ville naît de l'action d'un essaim d'hommes groupés, qui s'installe et agit sur un territoire. Deux formes de création peuvent être distinguées : les "agglomérations spontanées" et les "villes créées". Les villes spontanées se forment par l'action non formalisée du groupe humain sur le lieu qui lui permet d'exercer une activité et souvent aussi, de se défendre. La géographie a donc une influence essentielle sur la formation des villes spontanées. Ces villes se structurent plus ou moins bien, en fonction de la rapidité de leur croissance. C'est le mode de formation de nombreuses villes, mais c'est aussi celui des bidonvilles. La ville peut aussi naître de la volonté délibérée d'un groupe humain organisé. Ce sont les "villes créées", les villes neuves ou les villes nouvelles. Des hommes décident de créer une cité pour répondre à des objectifs liés à leurs activités. Les villes créées le sont par la volonté d'un groupe organisé, souvent très structuré, hiérarchisé avec un pouvoir fort. Elles le sont avec un objectif précis : colonisation, commerce, défense, industrie, etc. Le modèle le plus répandu est celui du plan en damier avec un quadrillage régulier de rues. Mais les villes neuves peuvent ne pas être totalement organisées et la fondation peut se limiter à la création de grandes structures sans que soit défini l'ensemble des quartiers qui se mettent en place progressivement au fur et à mesure de l'occupation des habitants.

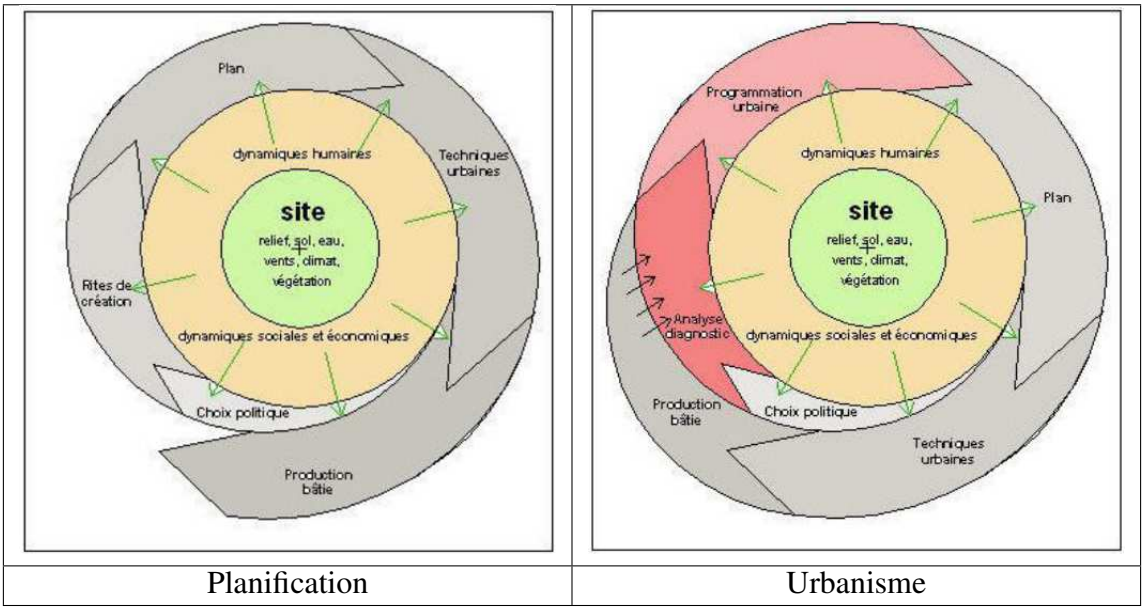
Développement des villes Après leur fondation, les villes se développent, plus ou moins vite. Certaines périclitent. Quel que soit leur origine (création ou naissance spontanée), leur développement associe presque toujours croissance spontanée issue d'une multitude d'initiatives autour d'une voie ou d'un axe et extensions programmées qu'elles soient d'initiative publique ou privée. En fait, la ville se constitue par stratification.

Reconstruction - transformation Les dynamiques individuelles et collectives des habitants conduisent à une permanente reconstruction de la ville sur elle-même. Une bonne partie des réglementations urbaines sont là pour gérer ces transformations du bâti en essayant de ménager l'intérêt collectif en canalisant les initiatives individuelles. Ces transformations peuvent prendre quatre formes : extension de l'emprise, surélévation, reconstruction ou dé-densification. La reconversion des friches est devenue un enjeu à la fois économique et symbolique des villes centres, face aux extensions périphériques et urbaines. On parle alors de reconstruire la ville sur la ville comme une alternative à l'extension urbaine.

Cycles de croissance urbaine Des sociologues ont développé un schéma radio-concentrique qui démontre une certaine superposition entre la répartition sociale et la répartition géographique [37] : le centre est occupé par un noyau constituant la zone d'affaires, et après une zone de transition on trouve un anneau de résidences occupées par les travailleurs les moins favorisés (ouvriers / immigrés, etc.) puis une zone résidentielle plus aisée avant des banlieues très chics.

2.3 De la ville à l'aire urbaine

La ville ne constitue plus aujourd'hui un ensemble compact bien défini, placé sur une commune unique. Elle forme une agglomération qui se développe sur plusieurs communes. Pour parler de ces ensembles composites qui s'agrègent autour des villes anciennes pour constituer un ensemble urbain, les géographes font appel à la notion d'aire urbaine. Pour l'INSEE (Institut National de la Statistique et des Études Économiques pour la France), une aire urbaine est un ensemble de communes, d'un seul tenant et sans enclave, constitué par un pôle urbain, et par des communes rurales et/ou une couronne périurbaine dont au moins 40 % de la population résidente dispose d'un emploi et travaille dans le pôle ou dans des communes attirées par celui-ci.



Planification | Urbanisme

FIGURE 2.1 – Comparaison entre planification des villes et urbanisme ; d'après [69].

2.4 L'urbanisme

Dans la suite de ce chapitre, les références historiques évoquées à titre d'illustrations concernent prioritairement la France lorsque l'échelle n'est pas précisée.

Définition D'après l'Encyclopédie Larousse[1], l'urbanisme peut être défini comme l'action réfléchie visant à disposer, à aménager ou à restructurer physiquement et socialement l'espace en vue d'assurer l'unification la plus harmonieuse et la plus efficace des fonctions que remplit un site donné. Il est inséparablement une théorie et une pratique dont l'exercice entraîne le recours à une technique. L'urbanisme intervient dans la disposition des bâtiments, la structure des réseaux de communication et des équipements publics, et, plus généralement, dans l'aménagement du territoire urbain.

En France, les premiers urbanistes sont apparus au tout début du 20ème siècle et cette appellation était alors essentiellement réservée aux architectes. D'ailleurs, LE CORBUSIER (architecte urbaniste, de son vrai nom Charles Édouart Jeanneret) soutient dans « Manière de penser l'urbanisme » publié en 1946 que « l'urbaniste n'est autre chose que l'architecte ». Toutefois, l'ouverture de l'urbanisme à la sociologie et à l'économie a progressivement permis de remettre en cause un tel formalisme. De nos jours, le métier d'urbaniste est exercé par des architectes, des ingénieurs, des administrateurs, des sociologues, des géographes et des économistes. L'urbaniste ne prend pas les décisions : il se situe au niveau de la conception et de l'organisation des transformations de l'ordre spatial dans le cadre de procédures administratives et de réglementations juridiques relatives à l'acquisition et à l'utilisation des sols.

De la planification des villes à l'urbanisme FRANÇOISE CHOAY, historienne des théories et des formes urbaines et architecturales, explique que les doctrines de l'urbanisme se fondent sur une analyse formalisée des besoins et des conditions de la réalisation urbaine [43]. Ce serait donc le caractère analytique des contraintes et des besoins qui distinguerait l'urbanisme des pratiques antérieures de planification des villes. Comme l'indique la Figure 2.1 , les principales différences entre la planification des villes et l'urbanisme sont la disparition des rites de création et l'introduction de deux phases majeures préalable au projet proprement dit : l'analyse de la situation existante qui doit conduire à un diagnostic et une phase de programmation urbaine qui peut être sommaire ou très élaborée. De plus, il y a désormais itération lorsque la croissance est

[1] http ://www.larousse.fr/encyclopedie/divers/urbanisme/100337

continue : une fois un morceau de ville produit, le site urbain est modifié et il faut reprendre le processus au stade de l'analyse et du diagnostic.

De l'utopie à la réalité La transformation de la société passe par l'invention de formes urbaines nouvelles. JEAN-BAPTISTE ANDRÉ GODIN, disciple de CHARLES FOURIER [2] et fabricant de poêles ayant fait fortune dans l'industrie, dessine les plans du Familistère de Guise [213] qu'il construit à proximité de son usine à partir de 1859 : une véritable ville dans la ville où vivront pendant près d'un siècle jusqu'à 2 000 personnes. Il s'agit d'un « palais social » qui vise à assurer le développement moral de l'ouvrier et de sa famille et dans lequel les ouvriers disposent d'un confort étonnant pour l'époque : "un allègement aux souffrances des classes ouvrières".

Des cités ouvrières aux lotissements populaires Pour éviter les problèmes d'éloignement à une époque où le transport individuel et collectif est inexistant, d'autres patrons pragmatiques construiront à proximité de leurs sites industriels des logements pour leurs employés les plus méritants. Mais ce modèle est rapidement critiqué car, en concentrant les ouvriers, le logement collectif devient "foyer d'immoralité" ou "phalanstère de la misère et du crime", selon certains contemporains. La maison individuelle paraît alors être la solution car elle permet d'éviter une trop grande concentration de familles ouvrières, politiquement et moralement dangereuse :

- elle permet à l'ouvrier d'améliorer son ordinaire grâce au jardin ;
- d'occuper ses heures de loisir de manière à éviter qu'il ne fréquente les cabarets ;
- de stabiliser la famille en le rendant propriétaire, donc responsable.

C'est ainsi que naquirent les premiers lotissements populaires.

Émergence du logement social C'est au cours du 20ème siècle que le logement social va se développer, avec l'apparition d'une exigence nouvelle qui va profondément modifier la ville et l'urbanisme : l'idée qu'il faut loger tout le monde dans des logements décents. La responsabilité de la création de ce logement va devenir, à partir des années 50 et leur transformation en HLM, une prérogative de l'État qui mettra en œuvre la construction de logement à grande échelle avec les grands ensembles.

2.5 Naissance de l'urbanisme moderne et applications pratiques

Concernant l'invention de modèles pour une ville adaptée au monde moderne, deux visions principales s'opposent :

- d'un côté, il y a ceux qui, pour pallier les maux de la ville industrielle, prônent la création d'une ville verte, plus proche de la nature : faire la ville à la campagne ;
- de l'autre, il y a ceux qui, fascinés par Manhattan, voient la métropole verticale comme un modèle qu'il s'agit de rationaliser et d'ordonner. Cette vision privilégie la ville en tant que creuset du développement économique au détriment de l'amélioration des conditions de vie de la population.

Évolution de la ville traditionnelle A côté de ces visions opposées, quelques architectes cherchent à marier densité, formes de la ville traditionnelle et conceptions hygiénistes. C'est le propos d'HENRI SAUVAGE, considéré comme l'un des principaux architectes français du premier tiers du 20ème siècle, qui propose de créer un nouveau type urbain : l'immeuble à Gradins [178]. A la place des jardins, les appartements sont dotés de vastes terrasses pouvant être plantées. L'îlot entier est pensé comme une vaste pyramide. Même si LE CORBUSIER conçoit lui aussi un projet d'immeubles à gradins, ce type sera un peu oublié après guerre,

[2] Philosophe français, à l'origne du concept de Phalanstère, un lieu de vie communautaire et harmonieux.

mais va réapparaître à la fin des années 60 et dans les années 70 avec les recherches innovantes pour un habitat intermédiaire.

Pendant la guerre, LE CORBUSIER va poursuivre ses réflexions en élaborant une doctrine nouvelle connue sous le nom de "Charte d'Athènes" [55] qu'il publiera ensuite sous le titre "La Ville fonctionnelle" en 1941. Il s'agit d'une théorie sans modèle qui propose des concepts simples, voire simplistes : "*le soleil, la verdure et l'espace sont les trois premiers matériaux de l'urbanisme*". Pas de modèle formel, mais des préceptes doctrinaires facilement applicables et facilement interprétables. La "Charte d'Athènes" prescrit un zoning strict, c'est-à-dire un classement des grandes fonctions urbaines par catégories. Seules les fonctions quotidiennes liées à l'habitation sont intégrées aux unités d'habitation. Ce document aura une très grande influence sur tout l'urbanisme d'après la seconde guerre. Même s'ils n'appliquent pas à la lettre ses préceptes, les grands ensembles en sont les fils directs, un peu déshérités, car bien souvent ils n'intègrent pas les équipements prévus, et la qualité des espaces verts, fondateurs du nouvel urbanisme, laisse à désirer.

Urbanisme moderne : quand les doctrines idéologiques se substituent à l'analyse scientifique Dans la pratique, la plupart des projets d'urbanisme ne s'appuient pas sur une connaissance approfondie des sites et des environnements sociaux sur lesquels ils vont agir. Ils substituent la phase d'analyse de l'existant et du diagnostic à des doctrines, qui tiennent lieu de théorie scientifique. Les fonctions urbaines sont réduites à quatre grandes catégories : *habiter, travailler, circuler, se cultiver*.

L'urbanisme moderne largement influencé par la "Charte d'Athènes" est conçu comme le moyen de faire une ville saine. Du point de vue de la composition urbaine, les réalisations s'appuient sur quelques principes majeurs :

- la volonté de créer un espace vert majeur au cœur de l'opération ;
- l'orientation des édifices en fonction de l'ensoleillement ;
- la ségrégation des circulations ;
- l'autonomie d'implantation des bâtiments par rapport à la voirie ;
- la volonté d'intégrer les équipements nécessaires à la vie quotidienne au parc ou au bâtiment.

Plus besoin d'analyse approfondie de la réalité puisqu'il faut raser l'existant, faire table rase pour reconstruire une ville nouvelle idéale.

2.6 Conséquences des politiques successives

Ensembles urbains figés, coûteux et difficiles à entretenir Aux objectifs directement liés à la doctrine de l'urbanisme moderne, l'État français en surimpose deux autres :

- résoudre la crise du logement ;
- créer une industrie du bâtiment digne de ce nom.

L'urbanisme moderne cherche à disposer d'un "sol libre" où l'on peut bâtir sans contrainte : l'immeuble est déterritorialisé et le sol n'appartient plus à ses habitants, mais seulement au gestionnaire, impersonnel ou collectif. Ce qui limite les possibilités d'évolution de la ville. Les formes urbaines de l'urbanisme moderne posent deux autres problèmes. Le premier est celui d'une extension extraordinaire des surfaces collectives qu'il faut aménager puis entretenir. Le second est celui d'une diminution des traitements minéraux au profit des espaces végétaux. Or un espace vert est beaucoup plus fragile qu'un espace urbain minéral et doit régulièrement être entretenu. Lorsque les réalisations sont de qualité, que les espaces verts sont entretenus, qu'il n'y a pas de problèmes sociaux, l'urbanisme moderne produit des résidences agréables où l'on habite dans la verdure. Sans doute l'urbanisme moderne, coûteux dans sa structure comme dans son entretien, aurait-il dû être réservé aux riches ?

Progression de l'étalement urbain et mobilité accrue C'est à partir du milieu des années 60, que les gouvernements, très critiques vis-à-vis de la production des grands ensembles, décident de mettre en place

une politique plus favorable aux maisons individuelles. Ce retournement politique, conjugué à l'enrichissement global de la population au cours des « trente glorieuses » (50-70) vont permettre aux classes moyennes de réaliser leur rêve : habiter une maison individuelle. Sous couvert de mieux répondre aux aspirations des familles françaises, cette nouvelle politique répond à plusieurs objectifs :

- mieux mobiliser les ressources financières des ménages, ce qui permet un certain désengagement de l'État à l'égard du financement du logement ;
- élargir le champ d'intervention et de profit des banques ;
- offrir de nouveaux débouchés aux grandes entreprises de construction, par le lancement de nouveaux produits leur permettant de rentabiliser leurs investissements dans l'industrialisation du bâtiment.

Ainsi, au lieu de se concentrer, les nouvelles extensions urbaines sont éclatées, augmentant considérablement les surfaces urbanisées. Les modes de vie changent avec comme caractéristique, celui d'un accroissement de la mobilité. On se déplace en permanence, et pas seulement pour le travail, mais aussi et de plus en plus pour le loisir.

Le naufrage des grands ensembles Au début des années 60, les conditions matérielles de vie dans les grands ensembles sont plutôt mauvaises dès qu'on sort de son logement bien équipé mais le climat social est bon. La dégradation du climat social des grands ensembles commence dans les années 70 et s'étend au début des années 80, à cause de trois facteurs principaux :

- la libéralisation de la construction de la maison individuelle qui conduit les classes moyennes à partir pour réaliser leur rêve ;
- la politique de regroupement familial qui conduit les familles des migrants à remplacer les classes moyennes dans les grands ensembles, alors que ce type de logement est totalement inadapté aux modes de vie d'étrangers qui sont des ruraux déracinés ;
- le chômage qui, à partir de la crise pétrolière de 1973, touche beaucoup les habitants des grands ensembles, en particulier les travailleurs immigrés, ce qui déstabilise leurs structures familiales.

Ceux qui restent sont ceux qui ne peuvent pas partir. Dès lors, on assiste à une concentration des problèmes. On constate une augmentation des loyers impayés qui conduit alors certains organismes HLM à minimiser l'entretien de leur patrimoine, ce qui induit une spirale de dégradation matérielle et symbolique du lieu. La construction des grands ensembles engendre un étalement urbain relativement limité, mais une forte fragmentation : les zones bâties sont rarement reliées à l'existant et constitue des enclaves séparées de la ville proprement dite. C'est d'ailleurs l'un de leurs problèmes majeurs.

Changement de politique : la rurbanisation, étalement urbain à grande échelle Dans une circulaire de 1973, OLIVIER GUICHARD alors en charge du ministère de l'Urbanisme, arrête la construction des grands ensembles. La nouvelle politique va engendrer un étalement urbain encore plus significatif avec une troisième poussée pavillonnaire prenant la forme de ce que BAUER et ROUX ont appelé justement la rurbanisation [19]. L'automobile diminue les distances, et désormais, une partie croissante de l'espace rural est utilisé par les urbains. La rurbanisation modifie profondément la structure de la ville et la vie des citadins qui, bien qu'ils soient urbains dans leur mode de vie, peuvent vivre à la campagne. C'est une nouvelle phase de transformation de la ville. Le parc d'habitat se modifie en profondeur pour atteindre, dans les années 90, 54 % de maisons individuelles contre 46 % de logements collectifs.

2.7 Théories urbaines après l'urbanisme moderne

Bilan critique de l'urbanisme moderne La volonté de rendre la ville claire et fonctionnelle conduit le modèle de l'urbanisme fonctionnaliste à supprimer la mixité fonctionnelle des quartiers alors que, lorsqu'on regarde l'histoire de la ville, tout laisse penser que c'est un fondement essentiel de la dynamique du fait urbain. Par ailleurs, elle conduit aussi à supprimer le découpage foncier qui est un autre élément matériel

fondateur de la ville. L'urbanisme moderne se réduit ainsi à la production d'espaces figés, sans réalité sociale et ne répondant pas aux aspirations évolutives des habitants. De tous côtés vont apparaître des propositions réhabilitant la ville traditionnelle qui cherchent à revenir à des formes urbaines plus classiques.

Redécouverte de la valeur de la ville ancienne et réhabilitation Les Italiens ont toujours préservé les centres anciens alors que les Français les éventraient pour remplacer les îlots par des tours et des barres. Ce seront les premiers à proposer de réhabiliter les villes pour améliorer les logements anciens plutôt que de les raser et reconstruire. Un certain nombre d'architectes-enseignants vont mettre au point des méthodes d'analyse du bâti existant dites "typo-morphologiques" qui consistent à regarder l'existant dans le détail soit pour le réhabiliter soit pour proposer des projets s'en inspirant. Il s'agit d'un urbanisme scientifique qui s'appuie sur les études, sur la connaissance du bâti mais aussi sur la connaissance des populations.

La défense de la ville européenne Dans les années 70-80, une proposition de retour à la ville classique européenne va se mettre en place. Avant, la diversité urbaine résultait de son évolution, de l'empilement progressif de ses couches. Maintenant, il faut créer de la diversité esthétique instantanée. Cet urbanisme nous propose une vision de la ville ancienne avec des décors pittoresques mais une fonctionnalité contemporaine, une ville idéale et humaine, comme elle n'a jamais existé. Aujourd'hui, beaucoup de promoteurs et certains élus ont adopté cette vision avec plus ou moins de talent et plus ou moins de cohérence.

Modernité et urbanité Dans les années 80, les architectes sont presque unanimes à critiquer les projets prolongeant l'urbanisme moderne. De leurs recherches ressort alors la volonté de réinventer un nouvel îlot qui s'inspire des expériences du début du siècle, avant l'urbanisme moderne : un îlot ouvert dans lequel le centre est occupé par un vaste jardin. Ce retour à l'architecture urbaine est largement répandu en Europe. Cela ne veut pas dire pour autant que les préceptes modernistes soient totalement abandonnés. En d'autres lieux, comme en Asie, en Chine, à Honk-Kong ou à Singapour par exemple, on construit encore de grands ensembles, avec des tours gigantesques qui empilent les fonctions et qui semblent bien fonctionner socialement.

La congestion urbaine et le troisième âge de la ville Parallèlement à ce qu'est devenu l'urbanisme officiel en Europe, réalisé avec plus ou moins de brio, deux doctrines se sont affirmées. D'un côté, il y a des architectes dans la mouvance du Hollandais REM KOOLHASS qui milite pour développer "la congestion urbaine". Son plan d'Euralille cherche notamment à renforcer la mixité avec la superposition des fonctions, même lorsque cela n'est pas « naturel ». De l'autre coté, il y a l'architecte et urbaniste français CHRISTIAN DE PORTZAMPARC qui prétend inventer "le troisième âge de la ville" [64], représentant une synthèse entre l'urbanisme moderne, aéré et vert et l'urbanisme traditionnel d'îlot. Il essaye de mettre en place des règles complexes permettant de concilier densité et végétal en évitant la monotonie de l'îlot HAUSSMANNIEN. C'est ainsi qu'OLIVIER MONGIN, écrivain français et directeur de la revue ESPRIT, n'hésite pas à appeler les architectes et urbanistes de tous bords à un véritable travail de couture, avançant l'hypothèse « *qu'avec le troisième âge de la ville se joue l'avenir de la démocratie, tout simplement* » [194].

Urbanisme durable et éco-quartiers Depuis quelques années, se développe ce qu'on appelle les "éco-quartiers" qui sont des projets urbains globaux à vocation écologique. Ce mouvement en plein développement est né en Allemagne et en Hollande avec quelques exemples pionniers en Autriche, en Grande Bretagne et dans les pays Scandinaves, et il a fini avec quelques décennies de retard par atteindre la France. Il n'y a pas de norme ni de recette d'éco-quartier. Leur forme et contenu varie en fonction des lieux et des enjeux. Tous les éco-quartiers n'ont pas les mêmes ambitions en terme d'économie énergétique, d'énergies renouvelables ou en terme de déplacements.

Urbanisme réglementaire et écologie Officiellement les PLU[3] devraient être attentifs à l'environnement. Dans les faits, l'interprétation de ce que doit être la protection de l'environnement varie et fait que les PLU ne sont pas toujours réellement écologiques. Par exemple, certains articles du règlement imposent des règles sur la hauteur ou l'aspect du bâtiment qui peuvent aller à l'encontre de la pose de panneaux solaires, de toitures végétalisées, etc.

2.8 Formes et structures de la ville contemporaine

Nous sommes aujourd'hui dans une civilisation urbaine où le modèle de croissance radioconcentrique reste le modèle de développement dominant des villes (le modèle « naturel »). Les rocades contribuent à cette croissance, tout en créant de nouveaux pôles péri-urbains. Les nouveaux lotissements cherchent avant tout une proximité avec les nœuds d'échanges du réseau. Les rocades génèrent ainsi des situations de périphérie en rupture avec une croissance continue de l'agglomération. C'est aux nœuds de ces voies (croisements, sorties) que s'implantent les nouvelles « centralités périphériques » que sont les zones d'activités, les centres commerciaux du grand commerce qui, au-delà de leur fonction de base, intègrent de nouvelles activités comme le travail tertiaire (bureaux) ou le loisir (multiplexes, centre de fitness, hôtels restaurants). Ce phénomène peut être redoublé par la création de plusieurs anneaux de rocades, comme c'est le cas à Rennes ou Paris. C'est le modèle français dominant.

Multiplication des types de tissus Avec l'éclatement urbain des années 1980-90, les formes urbaines se multiplient : nouveaux villages, lotissements périurbains, lotissements rurbains, zones pavillonnaires diffuses, villages ruraux, etc, ainsi que les nouvelles « centralités périphériques » : zones d'activités, zones artisanales, zones commerciales. Ces dernières sont désormais les principaux acteurs de l'économie commerçante en France : elles représentent 70 % des chiffres d'affaires contre 20 % dans les centres villes et 10 % dans les quartiers. La mutation touche aussi le centre ville qui d'un côté s'embourgeoise et de l'autre voit sa structure commerciale se transformer. Les commerces se spécialisent (prêt-à-porter) et il devient le lieu du développement d'annexes au grand commerce à travers le développement d'enseignes franchisées.

Sectorisation de la ville Le fonctionnement de ces fragments urbains diffère grandement de celui des tissus antérieurs : la plupart sont conçus comme des secteurs autonomes, desservis uniquement par l'automobile et dont les accès sont limités en nombre. On ne traverse pas ces secteurs comme on le faisait dans les tissus urbains traditionnels et leur dimension est généralement très supérieure à celle des îlots antérieurs.

Pour ceux qui peuvent choisir leur mode de vie, cette sectorisation peut paraître positive. On peut vivre entre soi dans un nouveau village, disposant à la fois du bien être de la campagne et, grâce à deux ou trois voitures, des services urbains. Mais pour les moins favorisés, ceux qui abandonnent le centre ville parce qu'il est trop cher, les choses sont moins positives. Déjà soumis aux nuisances induites par la voie rapide, ceux-ci se voient de surcroît écartés du centre-ville et de ses services par des infrastructures rendues quasi infranchissables.

Ces nouveaux types sont peu reliés aux autres secteurs et très figés dans leurs formes comme dans leurs programmes. DAVID MANGIN parle de sectorisation de la forme urbaine [180]. De plus, les règlements des lotissements interdisent toute activité (autre que les professions libérales), limitent fortement les extensions et interdisent les divisions parcellaires ou la construction en hauteur. Ces tissus urbains sont donc figés à l'exception de petits ajouts. De la même manière, bien qu'elles soient en permanentes transformations, les zones d'activités et les zones commerciales n'admettent pas de mixité : on ne peut y implanter de logements.

Dédensification La dédensification de l'agglomération est d'abord liée à l'intégration de territoires périphériques et rurbains dans l'aire urbaine. L'extension de la zone urbaine fait que désormais de nombreux

[3] Plan local d'urbanisme

secteurs très peu denses sont intégrés. La dédensification touche aussi tous les tissus de centre ville. La qualité de vie exige plus d'espaces verts avec des cours, des jardins et des logements plus grands. Pour le public, la densité a mauvaise presse et les électeurs font pression sur les élus pour que les règlements urbains minimisent les autorisations de construire. Toutes ces évolutions contribuent à faire baisser la densité des centres urbains. Les seuls secteurs où la densité augmente sont ceux en mutation, banlieues ou friches urbaines, qui lors du réaménagement sont construits alors qu'ils étaient souvent occupés par de grands bâtiments de faible densité.

Entrées de ville et ville franchisée En France, première nation touristique au monde, l'apparence et l'image de la ville ont une importance économique considérable. Toutefois, l'avènement du grand commerce, en deux décennies, va conduire à une homogénéisation globale des parcours d'entrée de ville, qui constitue une bien piètre vitrine pour la cité : les entrées des villes françaises sont largement composées de hangars décorés (bâtiment économique dont la façade est utilisée comme grand panneau d'affichage pour l'activité commerciale) alors que les canards (édifice ou enseigne géante qui épouse généralement la forme du produit vendu) sont beaucoup plus rares [275]. Cela relève de trois phénomènes à l'œuvre :

- celui d'une logique d'investissement à court terme : les bâtiments sont en perpétuelle évolution et sont amortis en quelques années, nous sommes dans la ville jetable ;
- celui d'une visibilité commerciale de la marque : logo, couleurs ou formes de bâtiment doivent être les mêmes sur tout le territoire, voire dans le monde entier (Auchan, Carrefour, Leclerc, Mc Donald, Ikéa, ...), les particularités locales doivent être gommées ;
- celui d'une visibilité locale du parking démontrant qu'il est simple de se garer.

De manière plus classique et plus conforme à la tradition urbaine européenne, les urbanistes français requalifient les entrées de villes, en cherchant à les domestiquer et à les rendre plus urbaines. Plusieurs voies ont été ouvertes pour cela :

- améliorations des aménagements urbains ;
- améliorations architecturales ;
- meilleure intégration du commerce d'entrée de ville dans la structure urbaine globale ;
- mutation du cadre législatif.

2.9 Conclusion

Pour OLIVIER MONGIN, « *nous vivons le temps des métropoles encerclées de banlieues explosives, quand ce n'est pas de bidonvilles menaçants, la ville devient alors une enclave protégeant des barbares* » [194]. L'un des enjeux majeurs de l'urbanisme de demain sera donc de parvenir à réhabiliter les quartiers dits « sensibles » et à transformer les bidonvilles en vraies villes.

Mais cette présentation montre toute la complexité intrinsèque de la structure d'une ville qui nécessite la mobilisation de nombreuses disciplines complémentaires et parfois opposées pour la comprendre : sociologie, économie, démographie, politique, histoire, écologie, etc.

La ville constitue un espace sensible en mouvement perpétuel dont il faut contrôler l'évolution en envisageant le long terme. Et de nos jours, penser sur le long terme, c'est intégrer la notion du développement durable (ou soutenable).

3

Développement durable

Le concept de développement durable, apparu il y a une vingtaine d'années et approuvé par 178 états lors de la première conférence de Rio en 1992, donne naissance à une conscience de la dégradation de l'environnement. Depuis, il a souvent été vidé de son contenu ou réinterprété selon les intérêts des forces en présence et du moment. Selon sa définition officielle (1987), le développement durable correspond à "*un développement qui répond aux besoins du présent, sans compromettre la capacité des générations futures à répondre aux leurs*". Il s'agit de proposer un changement de paradigme pour construire un monde économiquement efficace, socialement équitable et écologiquement soutenable. À ces trois piliers, s'est ensuite ajoutée la dimension culturelle [87].

3.1 Enjeux d'un développement soutenable sur le long terme

Les changements globaux Les changements globaux correspondent à l'ensemble des changements qui s'observent à l'échelle de la planète entière. Mais de nos jours, cette expression fait plus particulièrement référence aux effets de l'activité humaine sur l'environnement qui regroupent les nombreuses évolutions sociétales (évolution démographique, mode de vie, ...) et environnementales (climat, couche d'ozone, usage des sols, montée des océans, fonte des glaciers, biodiversité, ...). L'importance des changements globaux est liée à l'incertitude qu'ils font planer sur les générations futures ainsi que sur le devenir des écosystèmes dont elles dépendent. La complexité de ces changements résulte non seulement de leur dynamique spatiale et temporelle, mais plus encore de l'interdépendance des différentes problématiques. L'activité humaine dépasse depuis quelques décennies les capacités de l'environnement et, par conséquent, cette activité ne se maintient que par la destruction du "capital naturel" qui nous a été confié. Elle doit impérativement se réinscrire dans les limites de flux de matières premières et d'absorption de déchets autorisées par l'environnement [173]. Du fait de l'inertie des systèmes naturels, les prochaines années pourraient engager sur des siècles nos conditions de vie sur la Terre [82].

Deux exemples de l'obstination humaine La perspective d'une accélération de la hausse du niveau des océans au cours du siècle prochain, éventuellement accompagnée d'une augmentation de la force ou du nombre de tempêtes, laisse prévoir l'extension de submersions et l'exacerbation des reculs érosifs. Convient-il alors de s'obstiner à vouloir maintenir un état du littoral qui serait condamné par les lois de la Nature ? La solution de relocalisation des humains et de leurs activités se heurte à des mentalités collectives qui évoluent avec lenteur et perçoivent tout repli comme un renoncement [160].

Concernant la biodiversité, les projections pour le 21ème siècle font état de taux d'extinction d'environ dix mille fois supérieurs aux taux moyens du passé. Quels que soient les chiffres exacts, nous sommes en passe de connaître une crise majeure de la biodiversité. Mais cette fois ci, la cause est liée aux hommes qui ont réussi à bouleverser les équilibres naturels par le biais de la connaissance et de la technique. Le problème, c'est que cette espèce voit encore mal ce qui justifierait d'abandonner ce chemin [174].

De la croissance à tout prix Concernant le développement durable, il subsiste une ambiguïté liée à l'idée même que le développement est très souvent associé à une logique de croissance, contrairement à ceux qui revendiquent la fin de la croissance [87]. Les trois piliers identifiés lors du sommet de Rio (environnement, économique, social) étant posés équivalents et substituables, le monde des affaires peut justifier une diminution d'un pilier en contrepartie d'une augmentation sur un autre, le tout étant exprimé en valeur. Or, la conception marchande de la société et de la nature confère une valeur nulle à celle-ci : le poisson, l'eau ou la biomasse des sols n'ont pas de prix, leur seul prix apparent se réduisant à leur coût d'extraction. L'économie marchande dérégulée produit mécaniquement la ruine de l'environnement et la dégradation du monde vivant. Ne poursuivant aucun but autre que la croissance pour la croissance, le développement durable se réduit à une mise en bourse de la nature [285].

Des responsabilités partagées Les décideurs qui ont en charge de construire la société de demain doivent prendre en compte les changements globaux qui sont à l'œuvre pour assurer à nos sociétés un avenir sur le long terme. Or, la recherche scientifique nourrit parfois l'illusion que la course à la technologie émanant des travaux scientifiques peut proposer des issues à toute difficulté [82]. Sans prétendre aux miracles, il revient donc aux scientifiques de toute part de jouer un rôle essentiel pour décrire, comprendre, analyser, modéliser, prévoir et développer des stratégies adaptées à ces mutations. Dans ce contexte, l'interdisciplinarité est une aide majeure dans l'analyse et la compréhension des phénomènes et des systèmes complexes, dans le développement de technologies de rupture ou la compréhension et la conception de réponses aux grands enjeux sociétaux de la planète [26]. Alors, bien plus que de se côtoyer, les différents champs disciplinaires scientifiques de tous les horizons doivent absolument collaborer avec obstination, écoute réciproque et humilité pour contribuer à relever ces défis colossaux qui se dressent face à nous.

Détour philosophique et politique Pour Kant, seuls les êtres raisonnables ont une valeur intrinsèque et tout le reste est considéré comme moyen ou valeur instrumentale. Cette position "Anthropocentrique" ne reconnaît de dignité morale qu'aux seuls êtres humains. Le "Biocentrisme" considère lui que tout individu vivant est, à égalité avec tout autre, digne de considération morale. Certains environnementalistes vont plus loin avec l'approche "Écocentrique" en considérant qu'il faut accorder de la valeur non pas à des éléments séparés, mais à l'ensemble qu'ils forment [165].

D'un point de vue plus politique, l'affirmation qui laisse à penser que les sociétés humaines peuvent avoir un avenir infini dans un monde aux ressources limitées reste un pari hasardeux [202]. Et après plusieurs années de politiques qualifiées de développement durable, force est de constater la contradiction entre l'intention de la durabilité et la continuité de pratiques court-termistes [277]. Pour rebâtir les conditions d'un vrai développement durable, une refonte des régulations doit être opérée pour rendre très coûteux tout comportement dégradant les écosystèmes et très profitable ceux qui contribuent au maintien ou à l'amélioration des potentiels naturels [285]. L'objectif demeure le développement, mais encadré par une nouvelle technique de gouvernance, fondée sur la recherche d'un équilibre.

3.2 Dimensions urbaines du développement durable

Le modèle actuel de développement urbain des pays riches n'est pas durable. Si tous les pays se mettaient à consommer comme nous, les ressources seraient épuisées en quelques décennies. Le développement de la Chine et de l'Inde selon nos modèles de consommation est un scénario catastrophe.

L'enjeu de l'urbanisme en Europe aujourd'hui est celui de créer une ville durable. Pour cela, il est nécessaire de rendre plus dense et plus compacte la ville lors des mutations. C'est pourquoi on considère aujourd'hui que les friches sont devenues des enjeux majeurs pour « reconstruire la ville sur la ville ». Il faut donc veiller à ce que les projets soient denses, mais que cette densité soit acceptable pour le public. Il faut chercher à densifier le plus possible la ville en fonction du contexte et à recréer des liens entre les secteurs. Il faut redévelopper ce que DAVID MANGIN appelle la « ville passante » [180]. Mais, parallèlement, il faut transformer les règlements urbains qui, bien souvent interdisent la densité et les mutations urbaines, cherchant à préserver chaque fragment en l'état.

Métropolisation La métropolisation peut se comprendre comme l'avènement d'un réseau mondial d'aires métropolitains connectées entre elles par des réseaux d'échanges. Elle résulte de deux processus :

- la métamorphose du capitalisme qui s'universalise, entraînant des mutations profondes des modes d'organisation de la production parallèlement à la révolution numérique ;
- la polarisation spatiale croissante au profit des zones les plus développées.

A l'échelle locale, la métropolisation entraîne une restructuration spatiale et économique du marché du travail qui concerne un territoire plus vaste que la ville centre incluant les banlieues, le périurbain et parfois les espaces ruraux et naturels. Elle exige une grande mobilité de la part des individus pour effectuer les trajets entre les espaces résidentiels, les lieux de travail, des loisirs et des services. Dans un monde globalisé, seule la scène métropolitaine autorise le déploiement stratégique des contraintes économiques, sociales et environnementales relevant simultanément de l'échelle planétaire et locale. À condition de l'intégrer dans le territoire métropolitain, la ville représente donc un espace pertinent du développement durable en raison de sa pérennité historique, de son ancrage dans l'économie globalisée et de ses capacités d'adaptation face au changement climatique et des enjeux du mieux vivre ensemble [109].

Penser globalement / agir localement Les villes attirant l'essentiel des ressources et étant à l'origine, directement ou indirectement, de la plupart des émissions, la connaissance et la maîtrise de leur métabolisme [1] constituent des enjeux majeurs.

L'architecte RAYMOND LORENZO note qu'un habitant de la ville de Denver, dont la densité est très faible, consomme environ 3800 litres d'essence par an alors qu'un habitant de Manhattan en consomme 335 litres, soit onze fois moins. Cela veut donc dire que les formes de la ville, sa densité et les modes de vie qui leur sont liés ont une influence importante et donc qu'il est possible d'agir à travers l'urbanisme. *Il faut repenser les infrastructures urbaines, mais il faut aussi agir localement, en particulier sur la manière d'augmenter la densité par des aménagements favorisant un sentiment de faible densité : la présence végétale et les quartiers mixtes sont des éléments pouvant y contribuer.* Il faut avoir des visions globales en proposant des politiques spécifiques. Mais il faut ensuite que chaque agglomération ait une réflexion locale sur son impact territorial et sur les moyens de l'améliorer. Comme l'a souvent montré FRANÇOIS ASCHER, *on ne peut plus aujourd'hui faire l'urbanisme du haut vers le bas*, comme dans les années 60. Il faut écouter les habitants de base, faire du bottom / up, de la pédagogie, convaincre mais aussi écouter pour inventer de nouvelles solutions. Il faut donc avoir une action parallèle et forte à des échelles plus petites comme celle du quartier.

En fait, il s'agit d'une application du principe « penser globalement, agir localement ».

Mais le renouvellement du parc urbain représentant environ 1 % par an, il ne suffira pas à lui seul à produire une ville écologique. Il faut donc aussi réfléchir au moyen d'agir sur l'ancien, sur l'existant pour l'améliorer et le rendre plus performant.

Comprendre et agir sur la forme des villes [71]

[1] Ensemble des processus par lesquels les villes importent, consomment, transforment, stockent ou exportent énergies et matières.

Les modes de transport ont façonné les villes La forme des villes est en grande partie liée aux possibilités de déplacement des citadins. Les modes de transport rapide ont transformé la ville. Avec le chemin de fer, la vitesse moyenne des citadins est passée de 3 km/h à 15 km/h et les villes ont connu une première période d'étalement urbain caractérisée par des faubourgs et des banlieues disposées en « doigt de gant » le long des axes ferroviaires. Avec l'automobile, et le passage à une moyenne de 30 km/h, la ville a pu s'étendre à nouveau mais cette fois ci de façon dispersée : tout terrain étant devenu potentiellement urbanisable dans un large rayon autour des centres urbains. La ville peut alors s'étaler et s'émietter puisque les villages proches peuvent être le support d'une urbanisation souvent peu dense.

La vitesse a favorisé l'étalement urbain et les ségrégations La vitesse rend la proximité moins impérieuse et permet à de nombreux citadins de s'affranchir de la promiscuité. Avec la vitesse et des mécanismes fonciers qui renforcent souvent les ségrégations sociales et fonctionnelles, *la diversité dans la proximité* n'est plus le principe d'organisation de la ville, seulement celui de certains de ces quartiers. *La vitesse ne fait pas gagner de temps* : elle accroît les possibilités de choix et permet l'espacement.

Impacts des choix passés En raison de leurs valeurs patrimoniales, les politiques de mise en valeur des centres ont été plus fortes en Europe qu'en Amérique du Nord : leur dédensification a donc été moins marquée. Concernant les politiques foncières visant à limiter la consommation des sols, elles ont été historiquement puissantes dans un pays dense et vulnérable au risque d'inondation comme les Pays-Bas. Les fortes densités de l'« Europe rhénane » ont impliqué très tôt une promotion des transports publics alors que l'« Europe latine » y est restée longtemps indifférente, favorisant un système territorial fortement dépendant des transports motorisés et d'une énergie peu chère et abondante. En France, le secteur résidentiel ne compte que pour un quart de l'extension des surfaces urbanisées chaque année. Ce sont les infrastructures de transports, notamment routes et aéroports, les zones commerciales et les espaces d'activités qui sont responsables des trois quarts de l'extension spatiale.

Nécessité d'une approche systémique L'action sur la seule densité ne conduit pas nécessairement à réduire les besoins de déplacement. Sans action simultanée sur la localisation des diverses ressources territoriales, que ce soit les lieux d'emploi, de loisir, d'équipement ou encore d'habitat, la densification ne permet pas une réduction des distances pour relier ces différentes ressources. D'autre part, certains agencements périurbains ou ruraux peuvent se révéler favorables à la marche à pied, au vélo ou au transport collectif et les zones d'imbrication entre espaces urbanisés et territoires agricoles et naturels peuvent être intéressantes pour des circuits de proximité alimentaires et la préservation de la biodiversité. Enfin, bien que décrié, n'oublions pas que l'étalement urbain a permis à de nombreux citadins d'accéder à la propriété.

Une politique relative aux formes de la ville ne peut se résumer à une simple limitation de l'usage des sols, car agir sur la forme d'une ville, c'est agir sur les systèmes de formation de prix des fonciers, sur l'organisation des réseaux ou encore sur les mécanismes économiques qui la sous-tendent.

Écologie urbaine D'après le dictionnaire Le Robert, le mot écologie aurait deux sens :

1. étude des milieux où vivent les êtres vivants ainsi que des rapports de ces êtres entre eux et avec le milieu ;
2. mouvement visant à un meilleur équilibre entre l'homme et son environnement naturel, ainsi qu'à la protection de celui-ci.

L'écologie urbaine doit partir de l'étude du milieu en incluant non seulement les aspects physico-biologiques mais aussi les rapports sociaux humains, pour tendre vers un meilleur équilibre entre l'homme et son environnement naturel, ainsi qu'à la protection de celui-ci. L'écologie ne peut pas se limiter aujourd'hui à "la protection de la nature" mais doit avoir une vision plus large selon laquelle l'urbain est un milieu global et que la densité urbaine n'est pas en soi un mal, peut-être même une qualité. La ville intelligente, intéressée

à sa propre survie, doit être plus économe et attentive aux composantes de l'environnement, mais aussi aux équilibres sociaux qui, pour une part, la conditionnent.

Un aménagement écologique, c'est-à-dire répondant aux exigences de développement durable, doit porter sur une dizaine de thèmes [69] :

- mixité et densité ;
- qualité de l'air ;
- qualité de l'environnement sonore ;
- transport et déplacement ;
- gestion du cycle de l'eau (pluies, eaux usées, alimentation, nappe phréatiques, rivières) ;
- gestion des déchets ;
- énergies : consommer moins et tendre vers les énergies renouvelables ;
- présence de la nature en ville : bio diversité, milieu naturel, faune et flore ;
- paysage et protection du patrimoine ;
- une économie durable.

Éco-quartiers Les planificateurs et gestionnaires de la ville ont utilisé depuis les vingts dernières années différentes techniques pour limiter les émissions de polluants (en particulier les gaz à effet de serre) par les transports [253] :

- proximité d'un tramway ou d'un métro ;
- rapprochement des commerces et des écoles ;
- priorité donnée à la marche ou au vélo ;
- réduction de la place de la voiture.

Grâce à la mobilisation des citadins et à la contribution des élus et techniciens locaux, les éco-quartiers ont souvent dépassés les aspects environnementaux avec :

- une diversité du peuplement renforcée par une forte présence de logements ;
- l'intégration d'établissements pour personnes âgées favorisant l'intergénérationnel ;
- des lieux d'animation multiculturels et multifonctionnels.

Dans cette évolution liée aux éco-quartiers, l'enjeu est double :

- pousser le curseur écologique le plus loin possible ;
- faire en sorte que ce type d'approche ne soit pas réservé aux grandes opérations, mais imprègne toutes les opérations, même les plus petits lotissements.

Pour ROLAND CASTRO, architecte et urbaniste de renom habitué des plateaux de télévision et des stations radiaphoniques, les éco-quartiers posent effectivement la question des villes de demain. Mais au terme éco-quartier qu'il n'aime pas beaucoup, il préférerait qu'on emploie le terme quartier-éco, parce que l'objectif principal consiste d'abord à faire un quartier, un lieu que l'on habite, que l'on a envie de voir et qui nous sourit. Ensuite seulement, il s'agit de sublimer la contrainte lorsqu'elle arrive pour ne pas la subir. Dans notre cas, tout ne doit pas être subordonné à la contrainte écologique, ce qui pourrait alors conduire à faire très mal les morceaux de la ville. Proposer une série de logements, de maisons de diverses formes, de taille variable, mais à chaque fois dans un scénario de trajet, de rapport à l'autre, donner le sentiment qu'on peut être à la fois chez-soi et avec les autres si on le veut : c'est vers cela qu'il faut tendre. De nos jours, beaucoup d'architectes tentent de faire des logements qui ne soient pas alignés, des logements qui soient dans un rapport les uns avec les autres, qui participent à la formation des rues, des places ou des venelles. Des logements qui font les centres et qui fabriquent de la marche, du trajet, de la promenade. « *Il faut faire les quartiers comme un promeneur, comme un flâneur, et créer de la surprise !* ».

Renouvellement des grands ensembles Les pratiques actuelles en urbanisme de renouvellement des grands ensembles en France [1] se concentrent autour du principe d'une recomposition urbaine visant à :

- désenclaver les sites par la recomposition des rues, des parkings, la création de moyens de transports collectifs, l'introduction d'équipements publics ;
- permettre une certaine mixité par la démolition des bâtiments posant le plus de problèmes (généralement les immeubles les plus hauts, en particulier les tours) et la création de nouveaux bâtiments valorisant le foncier, généralement des logements en accession pour les classes un peu moins pauvres et des immeubles locatifs proposant un type de logement spécifique améliorant l'offre ;
- améliorer la qualité des lieux et les possibilités d'appropriation par les habitants grâce à la transformation des espaces de proximité des immeubles.

Deux instruments majeurs sont aujourd'hui utilisés :

- la démolition- reconstruction ;
- la résidentialisation.

La démolition - reconstruction Pendant longtemps, l'État est resté réticent à l'idée de démolir des bâtiments qui n'étaient pas toujours amortis. Ce n'est qu'à partir du milieu des années 90, que l'idée de démolition va s'imposer avec, pour objectif, de modifier l'image des quartiers et d'inverser les processus de dévalorisation. L'idée globale est que, pour modifier en profondeur ces quartiers, il faut recréer de la mixité sociale et que, pour cela, il faut favoriser l'intervention d'opérateurs privés dont la production s'adresse aux classes moyennes. On démolit donc beaucoup aujourd'hui et l'on reconstruit, soit sur place, soit en lisière des zones, soit parfois un peu plus loin, des logements selon des formes urbaines plus traditionnelles : des rues, des places, de petits immeubles avec un nombre limité de cages d'escalier et, généralement, des hauteurs limitées.

La résidentialisation L'action sur le bâti ne répond pas à tous les problèmes, mais elle est utile. C'est un des moyens de participer à une transformation positive du lieu. La résidentialisation est une forme particulière d'action qui s'oppose dans son concept à la démolition / reconstruction : on requalifie les espaces collectifs et les espaces publics, afin de transformer les formes urbaines issues des théories de l'urbanisme moderne. La résidentialisation vise à "configurer un ou plusieurs immeubles et leurs abords en une unité résidentielle clairement identifiée, matérialisée par un marquage plus ou moins symbolique de ses limites". Cette requalification des espaces de proximité, publics et privés des grands ensembles, permet de gérer le problème des poubelles, du nettoyage des espaces publics et privés. Mais aussi d'améliorer le confort du logement : on peut ajouter des balcons, créer des jardinets pour les logements du rez-de-chaussée. En modifiant le statut du sol et les formes urbaines, la résidentialisation répond aux besoins des habitants en créant des lieux bien différenciés et supprime l'indétermination des espaces du quotidien au-delà du logement.

3.3 Enjeux à l'export du savoir-faire français en matière de ville durable

Le Paris de Haussmann a été interprété dans nombre de grandes villes à travers le monde. A l'échelle de la planète, il est reconnu que Londres, New York, Paris et Tokyo sont observées et influencent les orientations des autres grandes villes qui sont le réceptacle des populations qui s'y agglutinent [41].

Compte tenu de ces éléments et forts de leur expérience en matière de ville durable, certains acteurs français se mobilisent et se regroupent pour exporter leur savoir faire. C'est le cas par exemple de VIVAPOLIS[2], une marque française de la ville durable qui cherche à fédérer de nombreux acteurs autour de quatre caractéristiques fortes qui définissent la « French touch » :

- *l'homme* positionné au cœur de chaque projet pour améliorer les conditions de vie de l'habitant ou de l'usager ;

[2] www.vivapolis.fr

- *la performance* pour atteindre une consommation réduite des ressources naturelles et une intégration optimale des fonctions de la ville ;
- *une gouvernance* forte et participative pour pouvoir organiser la ville, la projeter dans le futur, la financer, définir des règles et la faire fonctionner au quotidien ;
- *une adaptation* systématique au contexte local reposant plus sur une démarche ou une vision que sur un modèle tout fait : à chaque cas correspond une solution particulière en fonction du climat, de la géographie des lieux, de la culture, de son histoire.

La France possède plusieurs champions dans différents domaines :

- en construction : BOUYGUES, VINCI, EIFFAGE ;
- en mobilité : ALSTOM ;
- en énergie : EDF, GDF SUEZ ;
- en gestion des déchets ou de l'eau : VÉOLIA.

Il y a également en France une multitude de petites et moyennes entreprises qui sont leaders sur des niches du développement durable. L'objectif est d'arriver à une *alchimie de la ville durable* en regroupant différents acteurs français autour d'une vision unique pour être capable de l'exporter. La construction de la ville se fait partout, et plus particulièrement dans les pays émergents comme le Maroc, la Chine, le Brésil ou la Turquie. Il faut être capable de travailler ensemble pour apporter des réponses globales aux questions posées afin de réaliser un quartier ou une ville durable. Dans ce contexte, les collectivités locales françaises pourraient servir de vitrine à ce qui peut être proposé à l'export.

Autre exemple, la création du Groupement d'Intérêt Scientifique « Modélisation Urbaine » (GISMu)[3] qui s'est donné pour objectif de faire converger les besoins entre les mondes de la recherche, de l'ingénierie, de la maîtrise d'ouvrage et de la maîtrise d'œuvre dans le champ de la modélisation urbaine, depuis la gouvernance et la conception des villes jusqu'à leur exploitation [130]. Sa vocation consiste à promouvoir une approche plus intégrée de la modélisation urbaine en abordant les différentes échelles spatiales et temporelles, en intégrant un développement des villes plus durable, en étudiant différentes typologies de villes et en explorant des domaines novateurs (comme la modélisation des flux entrants et sortants de la ville). Cette entité regroupe à elle seule 73 membres dont 13 établissements, 12 entreprises, 38 unités de recherche et 10 pôles, agences ou autres organisations. Parmi ses nombreuses missions, on retrouve la promotion à l'exportation du savoir-faire français en matière de modélisation urbaine.

3.4 La voiture au cœur d'une nouvelle révolution industrielle ?

Autrefois un luxe, hier encore une commodité, la voiture est devenue une nécessité pour une large frange de la population sous l'impulsion d'un mode d'urbanisation qui lui est dévolu et qui s'est largement répandu. A cause de sa vitesse, des grandes distances qu'elle autorise, d'un très grand choix d'itinéraires et d'un accès presque continu le long des voies, la voiture éparpille la ville et, avec elle, la demande de livraisons ou de collectes. Elle favorise la diffusion de l'habitat, la décentralisation des activités ou la concentration de certains services. Par les effets de coupure des voies rapides dangereuses et bruyantes, elle divise le territoire et sépare plus qu'elle ne lie de grands pavés voisins. Et dès qu'une artère prend de l'importance, l'arrêt et la vie en bordure devenant de moins en moins confortable, son accès se limite à quelques points de plus en plus éloignés ou aléatoires. En conformant la ville à ses possibilités ou besoins, la demande de transport s'est dispersée loin de l'offre crédible de transport en commun et l'éparpillement croissant a disqualifié les modes de déplacement doux (marche à pied, vélo).

Cet outil toujours plus envahissant pousse à bitumer et à bétonner à grande échelle des surfaces toujours plus gigantesques pour réaliser des infrastructures, des voies de circulation et des aires de stationnement qui lui sont dédiés. Le transport routier de marchandises qui représente 1/3 des émissions de CO_2 en ville a su profiter des infrastructures développées pour la voiture sans supporter les dépenses liées aux dégradations

[3] http ://www.urban-modelling.org/

qu'il occasionne. Le système routier en croissance a dépassé un seuil au-delà duquel il est devenu contre-productif. Il produit de grandes injustices et complique toujours plus les politiques de redistribution des richesses, quand ceux qui subissent ses nuisances ne sont pas ceux qui profitent de ses commodités. Les populations les plus pauvres peuvent rester piégées dans les zones les plus pauvres, les plus polluées ou bruyantes et les plus privées d'emplois, de services urbains ou de paysages agréables [186].

Néanmoins, les jeunes occidentaux éprouvent de moins en moins d'intérêt pour l'automobile et ils n'ont plus envie de la grosse voiture de leurs parents. L'imaginaire dominant de notre univers automobile durant les trente glorieuses n'est plus adapté à cette nouvelle génération qu'il va falloir prendre en compte pour réinventer les moyens de déplacement de demain. L'automobile doit se redéfinir et on peut faire l'hypothèse de nouveaux modes de transports situés aux interstices des deux et quatre roues : probablement un modèle de petite voiture léger et écologique.

Mais pour le moment, des pays comme l'Inde ou la Chine sont dans une dynamique de rattrapage économique et encore dans la fascination pour un modèle occidental qui n'est pourtant pas viable. Chaque jour, 1 200 véhicules nouveaux sont immatriculés en Chine et la terre comptera cinq fois plus de voitures d'ici 2050. Ceci va provoquer une course aux matières premières de plus en plus effrénée au fil des années à venir, avec des conséquences déjà visibles (par exemple l'intérêt croissant de la Chine pour l'Afrique). Il va pourtant falloir adopter de nouveaux modes de croissance écologiquement responsables et le modèle énergivore apparu au début du 19ème siècle va devoir se réinventer. C'est une nouvelle révolution industrielle qui est en train de se dessiner sous nos yeux, et la voiture est au cœur de ce bouleversement [20].

3.5 Utopies

3.5.1 Une « Utopie Concrète » pour Paris

Le Grand Paris ROLAND CASTRO fait partie d'une génération d'architectes et urbanistes qui a énormément réfléchi et appris des échecs du passé, une génération qui a beaucoup déconstruit la pensée des temps modernes. Sa pensée intègre l'aspect politique et citoyen d'une architecture urbaine. Il est partisan de la réhabilitation par le remodelage des grands ensembles. Il a également initié le mouvement politique de l'*Utopie Concrète* pour transformer sans révolution la société vers plus d'égalité républicaine et de justice. Il y défend l'idée d'avancer des utopies concrètes et évolutionnaires pour redonner du sens à la politique. Pour lui, *l'utopie concrète : c'est la vérité de demain*. Tout comme les congés payés ou l'abolition de la peine de mort étaient des utopies en leurs temps, elles sont devenues des réalités de nos jours.

Il a été choisi par le président NICOLAS SARKOZY pour encadrer une équipe pluridisciplinaire lors de la consultation sur le Grand Paris : un projet de loi adopté le 19 juillet 2013 et visant à créer une métropole qui regroupera, dès 2016, Paris et trois départements de la petite couronne, doté de compétences en matière de logement et d'aménagement du territoire. Ce projet a été initié sous FRANÇOIS MITTERRAND[4], lancé par NICOLAS SARKOZY et repris sous FRANÇOIS HOLLANDE. Il s'agit donc d'un projet qui dépasse de loin les clivages politiques habituels de la France (gauche / droite).

Lorsque ROLAND CASTRO décrit ce projet, il considère un territoire réel plus grand que celui retenu par la loi, qu'il dit enfermée dans des limites administratives trop étroites. Pour lui, le Grand Paris s'étend sur une zone de 40 km par 40 km, concerne 10 millions d'habitants et regroupe 400 communes. Ce projet porte l'ambition de faire une métropole multipolaire caractérisée par une multitude d'éléments attractifs en dehors du centre historique. A cette échelle, le Grand Paris sera aussi grand que Berlin ou le Grand Londres. Le Grand Paris va permettre de faire la "ville à la campagne" avec un territoire beaucoup plus riche du point de vue végétal ou même agricole, un espace métropolitain dans lequel chaque quartier est attractif, sympathique, caractérisé par le plaisir d'y habiter et partagé par tous, ou chaque quartier « vaut bien » n'importe quel autre. C'est en cela que le Projet du Grand Paris représente une utopie concrète.

Mais en dehors des aspects techniques, ce projet est soumis à d'importantes crispations politiques liées

[4]Projet Banlieues 89 comportant un volet Grand Paris, confié à Roland Castro et Michel Cantal-Dupart en 1983.

d'une part à l'insuffisance d'une implication forte au plus haut niveau de l'état, et d'autre part à la présence de nombreux « barons » locaux qui disposent d'une influence importante et qui ne servent pas systématiquement la cause collective. Pour ROLAND CASTRO, le manque de budget identifié pour financer le projet n'est pas la question la plus grave car il y a beaucoup d'économies à réaliser en réformant des formes d'administration du territoire devenues superfétatoires avec le temps (notamment en supprimant les conseils généraux). De plus, le projet de loi relatif au Grand Paris prévoit :

- l'objectif de création d'un million d'emplois supplémentaires en Île-de-France sur les quinze prochaines années[5] ;
- l'obligation de construire 70 000 logements par an.

Paris ville-monde Dans sa note ouverte d'octobre 2013 adressée au Président de la République et intitulée « Vivre un Grand Paris » [41], Michel CANTAL-DUPART interpelle à sa manière François HOLLANDE pour lui faire part de son inquiétude face aux orientations de ce projet qui sont contraires aux convictions qu'il porte. Dans son document, il dresse un bilan complet de la situation et suggère sept objectifs pour redresser la barre.

Raccommoder les quartiers Il n'y aura pas de Grand Paris possible dans une ville à plusieurs vitesses : il ne s'agit pas de réaliser des opérations comme des éco-quartiers qui poussent de toute part sans se soucier de l'évolution des quartiers dans lesquels ils s'insèrent. Plus que de revisiter les Quartiers Prioritaires, il faut véritablement désenclaver les cités. Quand ils existent, les fleuves sont des éléments intéressants pour fédérer une ville car ils ne subissent pas les limites administratives.

Premier et dernier logement Les petits logements disponibles pour les jeunes (premier logement) sont souvent inaccessibles pour cause de loyer prohibitif et c'est la même chose pour un nombre important de retraités (dernier logement) qui doivent quitter la capitale parce que la vie et les logements y sont devenus trop chers : ce phénomène contribue à déséquilibrer la ville. Pour corriger ce phénomène, il est possible d'augmenter encore la densité de la ville dans des proportions raisonnables en utilisant ses réserves foncières aériennes qui sont vastes sans pour autant faire de grandes tours. D'autre part, il peut être intéressant de revisiter les immenses zones d'activités et leurs parkings pour proposer une évolution foncière lorsqu'il est possible d'y accueillir des logements. Au lieu de rajouter des bureaux déjà en nombre et parfois vacants, il faudrait leur préférer des programmes d'habitation pour arriver à un équilibre, notamment à proximité immédiate des gares pour inciter les habitants à utiliser les trains plutôt que des voitures (lorsqu'on construit des bureaux à proximité immédiate des gares, c'est l'effet inverse qui se produit : les gens quittent leur domicile avec leur véhicule et tentent par tous les moyens possibles de rejoindre leur lieu de travail en voiture).

Les chemins de l'écolier L'école est un lieu d'apprentissage de la ville, tant par le contenu de ses enseignements que par le chemin emprunté, itinéraire qui mène du domicile aux portes de l'école et fait découvrir aux écoliers les réseaux de la ville. Contrairement au parvis des mairies et des églises, on soigne rarement celui des écoles qui mériterait pourtant d'être plus accueillant pour recevoir les parents qui s'y regroupent en attendant leurs écoliers.

Cultiver le bien être Concernant l'implantation hospitalière, une position centrale dans l'agglomération peut être idéale à condition que les accès soient aisés et que soit prévue, pour le personnel soignant, une possibilité d'un habitat adapté de proximité. Si l'implantation est périphérique, l'accès à une intermodalité est impératif. Le plateau technique innovant et performant reste à l'hôpital, mais l'hôpital doit trouver des moyens de se déployer en ville au plus près du milieu de vie de ses patients, pour leur confort. Concernant

[5] D'après l'étude d'impact annexée au projet de loi (source : http ://www.senat.fr/rap/l09-366/l09-3667.html)

les possibilités de restauration des habitants et des visiteurs, la convivialité d'un Grand Paris passe par le bien manger. Paris est une ville magnifique, riche d'une poésie fabuleuse, mais qui gagnerait tout de même à offrir un meilleur accueil aux touristes venus du monde entier pour la rencontrer.

Donner accès à toute la ville Il est essentiel de pouvoir se déplacer dans toute la ville pour accéder au savoir, au travail, à la promenade ou aux biens de consommation. Les nouvelles lignes de métro express et les 70 gares et stations qui seront construites devront être l'occasion de valoriser les quartiers qui les accueilleront en recherchant l'intermodalité et la transversalité sans imposer un passage obligé par le centre de Paris.

Gommer l'attrait du vide Il faut délimiter de façon précise et sans tolérer de flou les parcelles urbanisées et urbanisables des parcelles à vocation agricole pour concevoir un urbanisme d'une certaine densité (sinon les lisières restent perméables pour les aménageurs) et créer un front de terre semblable à un front de mer pour offrir de larges perspectives aux paysages franciliens et protéger les terres agricoles qui peuvent fournir aux citadins des produits régionaux valorisants. Les terres agricoles ainsi protégées pourront évoluer sereinement pour mieux assurer les besoins alimentaires de la ville qu'elles côtoient.

Valoriser la Seine La Seine est la plus belle avenue de Paris, mais dès qu'on quitte le centre et qu'on arrive en banlieue, c'est fini ! Elle est pourtant un liant évident à tout le projet parce qu'elle coule à travers le territoire sans se soucier des limites administratives et parce que se concentrent sur ses rives universités et centres de recherche qui correspondent à l'image et à l'avenir du Grand Paris. De plus, elle abrite sur ses berges de nombreuses friches industrielles qui permettraient de qualifier un paysage exceptionnel.

3.5.2 La « Ville Creuse » : un exemple de ville idéale

La ville est le lieu complexe où de multiples forces s'opposent et se disputent. Comment par exemple réconcilier les attentes légitimes du citadin (plus d'espace, des facilités d'accès aux activités et services, un cadre de vie de qualité, convivial et sain, etc.) avec une réduction de nos consommations en ressources naturelles, une diminution de nos pollutions, de nos déchets ou d'un étalement urbain sans fin ? Existe-t-il une forme de ville optimale capable d'améliorer son fonctionnement, de réduire ses besoins, ses gaspillages et ses nuisances ? Y a-t-il une *forme d'organisation de l'espace* qui puisse nous conduire à ces différentes optimisations ?

JEAN-LOUIS MAUPU[6], bien qu'il ne soit pas urbaniste, s'est risqué sans état d'âme à cet exercice dans son livre intitulé « La ville creuse pour un urbanisme durable » [186]. La ville utopique qu'il propose repose sur trois grands principes complémentaires :

- un réseau de transport multiple qui suit une forme hexagonale constituée de 6 nœuds et formant une boucle (maille ou anneau) parsemée tous les 500 m d'une station de transport collectif (TC) ;
- des constructions avec une densité importante aux abords de la maille, son centre étant réservé à de grands espaces verts et naturels traversés d'allées, d'une rivière, d'un lac, d'exploitations agricoles, maraîchères et horticoles, de terrains de sport, d'un cimetière, de jardins et de vergers privés accessibles aux seuls modes doux et véhicules de service ;
- une mixité des lieux de logement et d'activité jusqu'à même les superposer dès que c'est possible.

Chaque anneau autonome fait environ 4 km de largeur pour 12 km de longueur, son tracé étant enveloppé d'un bâti mixte et dense pouvant accueillir 60 000 habitants dont 30 000 emplois et 25 000 ménages comptabilisant de 10 à 20 000 voitures. La ville peut croître en y ajoutant des anneaux supplémentaires sur le principe d'une structure alvéolaire en nid d'abeilles.

Sur le contour de chaque anneau, on retrouve une ligne de TC aérienne guidée qui comprend 24 stations ainsi qu'une voie dédiée aux vélos et piétons, et une troisième ligne si possible en sous-sol réservée aux

[6]Ingénieur Arts et Métiers, chercheur à l'Inrets : Institut National de Recherche sur les Transports et leur Sécurité.

véhicules utilitaires, aux véhicules de service ou de transport de marchandises ainsi qu'aux voitures particulières qui se partagent l'espace du niveau inférieur également réservé aux machineries, aux divers réseaux techniques, aux aires de stockage et à la totalité des parkings publiques et privés. Les activités engendrant les plus gros trafics de marchandises se situent près ou au niveau de l'artère routière alors que celles qui induisent un trafic de voyageurs sont positionnées à proximité de la ligne de TC. les lieux de logement et d'activité sont mélangés et répartis selon leur nécessité en termes de proximité des diverses autres lieux, de la lumière, de l'air libre ou en direction de la verdure, tout cela pour réduire les besoins de déplacement, la longueur des trajets et respecter la diversité des attentes.

Toutes les constructions réparties de façon homogène étant à proximité immédiate de la boucle, et plus particulièrement des stations de TC, la ligne de TC devient très attractive et la plupart du temps, la voiture n'est plus d'une nécessité absolue. De plus, comme le réseau de TC dessert l'ensemble des lieux fréquentés sur l'anneau, il est en mesure de capter 1 à 2 voyages par habitant et par jour contre 0,1 à 0,2 en règle général. Compte tenu de sa structure en boucle, la ligne de TC peut donc être rentabilisée avec 10 fois moins d'habitants desservis. Pour mieux répondre à la diversité des attentes individuelles, des pavillons, villas et cabanons peuvent être tolérés dans l'espace de verdure en conservant des règles justes et strictes d'accessibilité, d'implantation et de construction pour garder à la partie dense la vue et la proximité de la verdure et des beaux espaces libres.

Pour l'approvisionnement en marchandises, les transporteurs déposent ou prennent leur chargement sur des plateformes logistiques situées aux entrées de la ville et directement branchées sur les voies interurbaines rapides ou sur une gare ferroviaire. Un service de véhicule adapté peut alors distribuer et collecter les colis ou assurer les échanges internes en accédant aux points de livraison et de collecte localisés sur le court linéaire de voirie couvert qui reste d'accès facile aux poids lourds.

Pour traverser un anneau (diamétralement), il faut compter entre 40 et 60 mn à pieds, 15 mn à vélo et 15 mn en TC. Une ville constituée de 3 hexagones formant un trèfle équivaut à 2,5 anneaux bâtis, soit 150 000 habitants. Un motif de 7 hexagones formant une fleur à 6 pétales équivalent à 5 anneaux, soit une ville de 300 000 habitants. Avec une bande urbanisée de 500 m de large sur une maille complète, on obtient une densité de 100 hab/ha[7] (carré de 500 m de cotés avec une station de TC en son centre, soit 25 ha avec 2 500 hab) où chaque habitant est à moins de 350 m d'une station de TC et à moins de 250 m d'un grand espace de verdure. Avec une bande urbanisée de 1 000 m de large, la densité tombe à 50 hab/ha [8] et chacun est à moins de 550 m d'une station de TC. Le grand espace de verdure central devient moins accessible (il reste à moins de 500 m) mais il est davantage présent dans toute la bande construite.

Avec cette organisation, la ville creuse permet de réduire efficacement les besoins de mobilité et de transport de biens, elle se caractérise par un trafic concentré sur sa boucle et une forme urbaine attractive (densité pas trop élevée, logements individuels, proximité de la nature) économe en énergie et matériaux, produisant peu de déchets et s'approvisionnant en priorité sur son territoire agricole. Elle peut s'accommoder facilement de la géographie d'un lieu, de sa géologie ou de son histoire et s'y insérer en douceur tout en le préservant, le tracé choisi pouvant suivre un chemin existant, encercler un petit bourg de caractère ou l'intégrer comme nœud principal.

Une ville ainsi constituée pourra s'adapter à des temps plus difficiles, en particulier à une mobilité bien plus coûteuse des biens et des personnes. Pour s'en persuader, récapitulons ici les attraits marquants de cette ville pour un développement urbain durable, *pour être riche de ce dont on n'a plus besoin* :

- la structure compacte et dense du bâti favorise les économies (en matériaux de construction, d'énergie de chauffage ou de climatisation, en eau, en linéaire et raccordement des réseaux techniques) ;
- la possibilité de circuler en zone couverte et continue favorise la marche à pied et l'usage du vélo ;
- la proximité des lieux (services, formation, travail) réduit la longueur, la durée et le coût des déplacements contraints ;
- la surface occupée par le mode routier, parkings compris, est réduite (5 % de la surface de la maille) ;

[7] Densité proche de celle d'une cité jardin comprise entre 160 et 200 hab/ha
[8] Correspond à une densité proche d'une zone pavillonnaire

- tout point du bâti principal est à proximité immédiate d'une ligne de TC ;
- continuité de service des TC (deux itinéraires pour se rendre d'un point à l'autre) ;
- proximité constante des espaces verts et de toutes les activités récréatives rendant la densité viable ;
- absence de nuisances liées à la circulation automobile : effet de coupure, bruit, pollution, danger, bouchon ;
- possibilité d'extension de la ville par ajout de maille (reproduction du motif) sans remise en cause ou dégradation des conditions existantes ;
- extrême simplicité grâce à une structure annulaire (flux de marchandises entrant et sortant faciles à rationaliser, moins de carrefours, signalisation simplifiée, séparation des modes de transport, des flux en eau, ordure, électricité ou de chaleur) ;
- plus d'éventration permanente des voiries par des tranchées ;
- facilité de maintenance des réseaux (bonne accessibilité, détection des fuites de fluides) ;
- pas d'extrémités de ligne de TC pauvres en trafic, pas de tronçon central surchargé.

JEAN-LOUIS MAUPU nous propose une armature de ville sans centre ni banlieue, une nouvelle *machine à habiter* organisée tout simplement le long d'une ligne optimale de transport collectif. Mais cela nécessite une autorité organisatrice indépendante (ou une auto-organisation durable) capable de la projeter, de la bâtir et de la maintenir, capable aussi de transcender les contradictions de son opinion publique, et c'est probablement ce dernier aspect qui constitue la principale utopie de son projet.

3.6 Conclusion

L'urbanisme moderne n'est pas durable car il crée des situations figées, non évolutives. De plus, la réalité de l'évolution sociale des grands ensembles français a rappelé qu'on ne pouvait omettre les dimensions anthropologiques et sociologiques de la ville et de l'habitat et qu'au-delà de la satisfaction de ses besoins de base (manger, s'abriter), l'homme est un être social dont les besoins varient en fonction des cultures et des organisations sociales. Il faut donc *penser globalement tout en agissant localement*.

L'un des enjeux majeurs d'un urbanisme durable est de redonner une certaine compacité à l'agglomération, donc à augmenter la densité de ses secteurs périphériques. Pour y parvenir, il faut sans doute changer les réglementations urbaines en imposant des densités minimales pour limiter la croissance urbaine et permettre de doter l'agglomération de services performants et plus économiques. Une densité suffisante peut être retrouvée en reconstruisant la ville sur la ville. L'introduction de la nature dans l'urbain est un facteur déterminant pour rendre cette densité supportable. Dans ce contexte, il parait également nécessaire de redonner à la voiture la place qu'elle mérite en ville, ni plus ni moins, et d'imaginer des transports collectifs accessibles à tous et performants. D'autre part, la mixité sociale et fonctionnelle apparaît essentielle pour limiter nos besoins en déplacements, et par conséquent limiter nos consommations d'énergies et diminuer les nombreuses pollutions (qualité de l'air, bruit, danger, etc.) tout en assurant l'équilibre et la cohésion de ces grands ensembles vivants.

Le développement de nos villes déterminera l'avenir de l'Europe[9] [108].

Dans la pratique, il faut néanmoins distinguer deux situations : celle des pays développés qui possèdent déjà un milieu très urbain et relativement stable et celle des pays en développement qui voient leurs paysages urbains se métamorphoser sous l'impulsion de populations rurales nombreuses qui viennent s'agglutiner en périphérie des villes ou dans les bidonvilles.

"*Des solutions en matière de ville durable existent, mais on ne sait pas les rendre désirables. La ville appelle de nouveaux récits, mais trop peu de personnes sont capables de les produire*" [20]. Pour mettre en œuvre ces solutions qui existent en matière de ville durable et les rendre accessibles au plus grand nombre à défaut de les rendre désirables, de nombreux projets de recherche ont vu le jour ces dernières années pour proposer des modèles et outils d'aide à la décision dans bien des domaines de l'urbanisme et accompagner ainsi les décideurs dans des choix éclairés. C'est notamment le cas de ces travaux de thèse qui contribuent

[9] Johannes Hahn, Membre de la Commission européenne en charge de la politique régionale.

au projet de recherche SUSTAINS pour proposer une aide à la décision opérationnelle destinée à la pré-programmation d'environnements urbains durables.

4

Cadre méthodologique et problématique

Dans un premier temps, nous présentons le cadre méthodologique dans lequel s'inscrivent nos travaux en revenant sur les contributions notables de l'urbanisme au développement durable et en présentant les grandes lignes du projet de recherche SUSTAINS auquel nous participons. Il s'agit d'un projet de recherche pluridisciplinaire qui propose de réaliser un outil d'aide à la décision pour la pré-programmation urbaine (4.1.2.1). En nous associant à ce projet qui regroupe des experts de différents domaines (urbanistes, architectes, statisticiens, concepteur d'interfaces homme machine ou de simulations 3D, chercheurs, ...), nous ancrons nos travaux de recherche dans la réalité du terrain en profitant de compétences croisées et complémentaires sur un domaine reconnu pour être complexe et très difficile à appréhender dans sa globalité. Ce sera également un bon moyen de valider nos modèles et résultats par des professionnels du domaine.

A partir de ces éléments, nous décrivons notre problématique en détail en faisant l'inventaire des problèmes de recherche que nous souhaitons adresser dans cette thèse. Après avoir évoqué les principaux concepts et théories associés à ces problèmes, nous les déclinons en questions de recherche précises et faisons une série d'hypothèses pour approcher des réponses présumées. Enfin, nous décrivons notre méthode générale de travail pour y parvenir.

4.1 Synthèse des éléments contextuels

4.1.1 Articulations entre urbanisme et développement durable

La contribution de l'urbanisme au développement durable est une idée largement répandue qui s'inscrit en France dans la loi de programmation relative au Grenelle de l'environnement avec une obligation pour les collectivités publiques, de par leurs prévisions et leurs décisions d'utilisation de l'espace, de réduire à la fois les émissions de gaz à effet de serre (GES) et les consommations d'énergies, notamment les ressources fossiles [70]. Pour répondre à ces enjeux, les urbanistes préconisent différentes orientations :

- limitation de l'étalement urbain ;
- promotion des transports collectifs et de la marche à pied ;
- meilleure répartition des activités et des logements dans l'espace.

Malgré ces recommandations apparues dès les années 70, la ville a évolué de manière assez éloignée de ces grands principes. Certaines études [185, 196, 209, 11, 262, 287, 36] contestent même l'efficacité de l'urbanisme dans ces domaines, en évoquant :

- des impacts limités en comparaison de ceux attendus dans les domaines de l'automobile (amélioration

des moteurs) et du bâtiment (meilleure isolation) ;
- des coûts de mise en œuvre très importants ;
- des effets perçus sur le long terme inadaptés face à l'engagement de diminuer, en France, les émissions de gaz à effets de serre par quatre d'ici à 2050.

Néanmoins, les modes d'organisation spatiale des villes, et notamment la densité, ne sont pas neutres du point de vue des émissions de gaz à effet de serre. Il a été établi que, selon que l'on vit dans une zone bien desservie et bien équipée, ou que l'on vit dans une zone peu dense et dépendante de l'automobile, la consommation d'énergie pour la mobilité varie dans un rapport de 1 à 3 pour des personnes comparables en termes de niveau de vie et d'âge. Ces variations sont principalement imputables aux distances parcourues pour réaliser les activités quotidiennes. Les habitants des cœurs urbains réalisent une part plus importante de leurs déplacements à pied ou en transport en commun, mais le gain obtenu par les habitants des zones denses peut être contrebalancé par des déplacements saisonniers ou de fin de semaine plus importants, l'accès à un jardin réduisant la mobilité de loisir grâce à « *l'effet barbecue* »[1].

Outre la densité, de nombreuses études montrent que la répartition des ressources (logements, emplois, services, etc.) à l'intérieur de l'espace urbain joue un rôle très important sur la consommation énergétique et l'émission de GES du fait des déplacements. Le levier d'action principal est moins la forme urbaine ou la densité que la manière dont s'agencent les individus, activités et réseaux de transport pour faciliter une accessibilité peu émettrice de GES et peu consommatrice d'énergie.

Enfin, certains supposent qu'un prix élevé de l'énergie pourrait avoir un impact beaucoup plus efficace que l'urbanisme et l'aménagement du territoire pour réduire les émissions de GES. Mais jouer sur le seul effet du prix conduirait à faire l'impasse sur les conséquences sociales de telles mesures, les ménages pauvres situés principalement dans les espaces périurbains à faible densité étant parmi les plus vulnérables face à une augmentation du coût de l'énergie. Pour assurer l'équité sociale inscrite dans les principes du développement durable, il est nécessaire de prévoir un aménagement urbain qui autorise au plus grand nombre des déplacements quotidiens indispensables à la vie économique, sociale et culturelle de toutes les villes, même en cas d'usage rare ou cher de l'énergie.

En complément des réductions de GES et des consommations d'énergies, le renforcement des liaisons piétonnières, l'amélioration des réseaux de transport collectifs, une plus grande imbrication des logements et des activités, etc. contribuent à la qualité urbaine dans ses différentes dimensions (sociale, environnementale et économique). Ces principes d'aménagement jouent notamment un rôle majeur sur l'acceptabilité sociale de la transition énergétique en marche.

Dans le cadre de nos travaux, nous nous intéressons plus particulièrement à une organisation de l'espace urbain (occupation des sols) qui puisse satisfaire les objectifs suivants :

- densité maîtrisée ;
- proximité de la nature ;
- mixité sociale et fonctionnelle ;
- viabilité économique des projets urbains ;
- aménagement à l'échelle d'une ville nouvelle ou pour le renouvellement complet d'un quartier ;
- bien être des habitants (en évitant le bruit, en favorisant les modes de transport doux, . . .).

4.1.2 Un champ d'application moteur de la recherche théorique

Les travaux de recherche relatifs à cette thèse s'inscrivent dans le cadre du projet de recherche national FUI[2] SUSTAINS (http ://www.sustains.fr/). Ce projet propose de réaliser un outil d'aide à la décision pour la préprogrammation urbaine (cf. définition dans la section ci-dessous) et le choix des systèmes énergétiques. Cet outil doit permettre d'appréhender la complexité des modèles urbains (résidentiels, industriels, services pu-

[1] Dans un espace périurbain, on est généralement de gros consommateurs de kilomètres automobiles pour aller travailler, mais le week-end, on a tendance à rester chez soi pour profiter du jardin et faire un barbecue.

[2] Fonds Unique Interministériel.

blics) dans ses dimensions sociales, économiques, énergétiques, de mobilité et de durabilité. L'intégration, la visualisation et la manipulation de ces dimensions dans une plateforme informatique[3] opérationnelle de la ville visent à replacer les différents acteurs (élus, financeurs, société civile) au sein du processus de décision en se focalisant sur deux enjeux : environnement et énergie.

Ce projet, commencé en 2011, s'est achevé le 11 février 2014 lors d'une présentation des principales avancées scientifiques et résultats obtenus en présence de tous les acteurs impliqués, des différents financeurs et représentants de structures associées au développement urbain.

4.1.2.1 Pré-programmation urbaine

Face à la nécessité d'opérer des choix de développement urbain qui s'inscrivent dans une perspective durable, les collectivités sont amenées à réaliser des arbitrages complexes, en mesurant a priori les conséquences environnementales et économiques des solutions envisagées.

Pour être accompagné dans ces choix difficiles, il est désormais essentiel d'utiliser une méthodologie et un outil permettant de modéliser la programmation d'environnements urbains dans une phase amont, aussi bien pour la création de villes nouvelles que pour la restructuration urbaine. Cette phase constitue la pré-programmation urbaine (PPU), étape préalable au lancement d'une étude urbaine complète. Elle permet d'établir dans une démarche systémique les éléments de programmation (proportion de maison d'habitation, immeuble collectif, bureau, zone artisanale ou industrielle, équipement, espace vert, etc.) essentiels à intégrer dans la conception d'une ville durable. Elle fournit sur une emprise territoriale donnée des études de capacité d'accueil d'un ensemble urbain avec tous les éléments de programmation y compris les équipements publics induits à partir d'un taux d'emploi spécifié.

La pré-programmation comprend le positionnement des centralités et axes urbains, la répartition des intensités (niveaux de densité urbaine, de mixité fonctionnelle, de mobilité) et des espaces de respiration, l'organisation en quartiers et en îlots, le tout en fonction d'une population donnée, d'un taux d'emploi, d'une emprise territoriale et d'un ensemble de descripteurs tels l'équilibre des aménagements logements/emplois, la diversification des surfaces ou la création d'espaces de vie collectifs polyvalents.

4.1.2.2 Description générale du projet

Le projet SUSTAINS, réalisé en collaboration étroite entre des praticiens de la ville, des experts du domaine de la production et distribution énergétique, et des chercheurs, consiste à :

- concevoir un **modèle théorique de développement urbain durable** reposant sur l'expérience pluridisciplinaire de praticiens de l'aménagement urbain (Établissements Publics d'Aménagement de Marne-la-vallée : *ÉPAMARNE*). Ce modèle de pré-programmation urbaine est construit autour de la notion d'intensités urbaines intégrant niveau de densité, mixité fonctionnelle et support de mobilités.
- étudier et réaliser **un modèle informatique interactif** qui génère une planification urbaine à partir des règles du modèle théorique de développement urbain durable et de contraintes posées sur les indicateurs (par exemple le maintien d'indicateurs dans des fourchettes de valeurs). Ce modèle doit permettre une visualisation de la ville et de ses indicateurs, ainsi qu'une interaction avec celle-ci (par exemple modification des seuils ou déplacement des éléments de programmation). Cette démarche inverse de conception consiste en (i) l'expression sous forme de contraintes des propriétés désirées de la ville au travers des nombreux indicateurs qui la caractérisent (habitat, énergie, transport, services), (ii) la génération d'une planification urbaine par résolution de ces contraintes (création d'un maillage routier, identification des quartiers et des îlots, caractérisation de la nature des ilôts) et (iii) l'interaction avec cette planification à différents niveaux (répartition spatiale, agencement des quartiers, seuillage des indicateurs) en maintenant la satisfaction des contraintes et la cohérence de l'ensemble.
- concevoir et développer un module de **simulation et d'optimisation des schémas énergétiques**. Ce module permet de modéliser les besoins énergétiques d'un ensemble urbain (zones résidentielles,

[3] On utilisera le terme de « plateforme » par ailleurs dans ce document pour faire référence à cette suite logicielle.

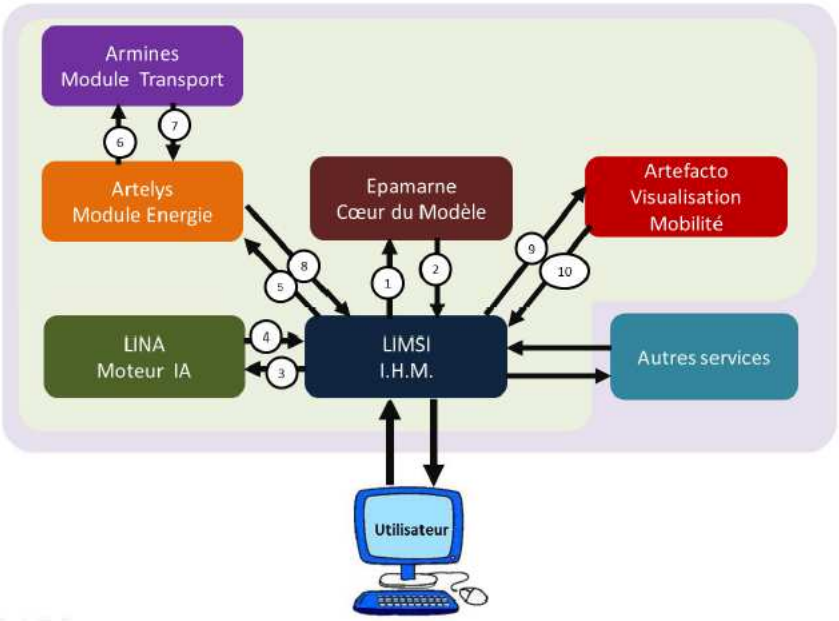


FIGURE 4.1 – Échanges entre les modules attribués aux différents acteurs du projet SUSTAINS : ① Surface à aménager et autres paramètres ② Objectif en nombre d'îlots de chaque type (maisons, immeubles d'habitation, commerces, écoles, bureaux, ...) ③ Adresse du fichier de configuration ④ Contenu du fichier de configuration dans lequel est assigné à chaque îlot symbolique une fonction urbaine (maisons, immeubles d'habitation, commerces, écoles, bureaux, ...) ⑤ Données sous la forme d'un fichier KML ⑥ Entrées issues du traitement relatif au fichier KML ⑦ Volumes annuels de consommation par mode sous la forme d'un fichier CSV ⑧ Fichier KML dans lequel a été inclus l'ensemble des résultats énergie et transport ⑨ Fichier 3D de base ⑩ État du traitement du fichier 3D et index du projet. (Schéma extrait du rapport de fin de projet SUSTAINS.)

industries, services publics), de simuler les profils de production/distribution pour répondre à cette demande et d'évaluer l'impact environnemental ainsi que les coûts complets correspondants.

- expérimenter et **valider le modèle à différentes échelles de villes** (éco-quartier, ville,...) en utilisant les données socio-économiques, écologiques et d'aménagement fournies par des projets en cours menés par *ÉPAMARNE*.

La figure 4.1 présente les modules répartis entre les différents acteurs impliqués dans le projet SUSTAINS ainsi que les flux d'échanges qui viennent constituer et enrichir une pré-programmation urbaine.

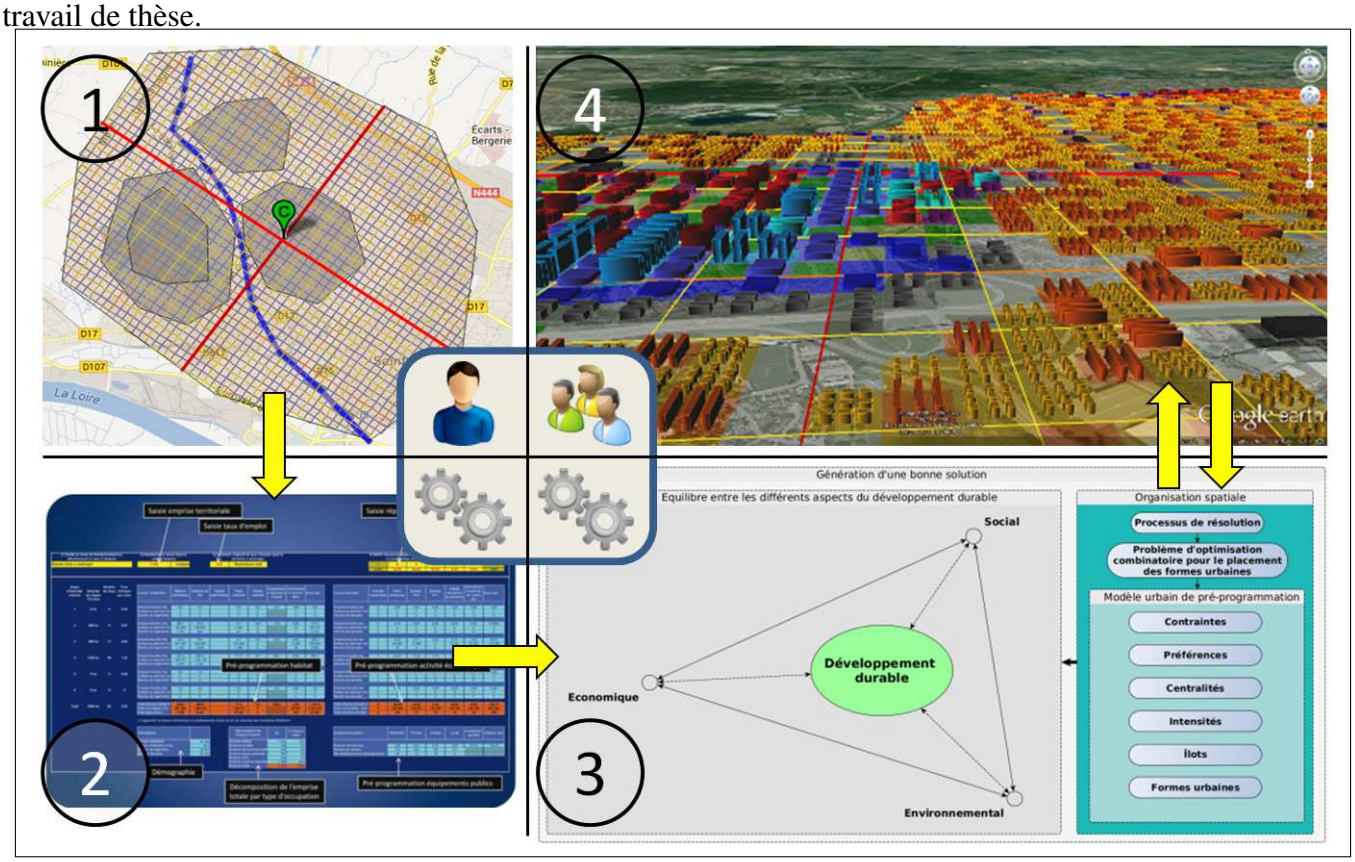
Cette thèse coïncide avec le module « *LINA / Moteur IA*[4] » de la figure 4.1 qui a la charge de produire un **modèle informatique interactif** (cf. section "Description générale du projet" ci-dessus).

4.2 Le cadre méthodologique

Notre travail s'intéresse d'une part aux contributions notables de l'urbanisme au développement durable et se positionne d'autre part au cœur du projet de recherche SUSTAINS. La figure 4.2 illustre de façon synthétique le cadre méthodologique qui s'articule autour d'une plateforme globale d'aide à la décision pour la conception d'environnements urbains durables :

[4]Moteur IA pour "Moteur d'Intelligence Artificielle" : ce module informatique apporte l'intelligence nécessaire (*artificielle*) pour répartir au mieux et de façon automatique les éléments urbains sur la ville, tout en proposant une interaction utilisateur nécessaire à l'obtention d'une solution de compromis entre les décideurs.

FIGURE 4.2 – Conception d'un environnement urbain durable en 4 étapes : (1) paramétrage des contours du territoire urbain, des propriétés, des axes principaux, des centralités et des zones d'intensités ; (2) calcul du nombre de formes urbaines par niveau d'intensité ; (3) positionnement automatique des formes urbaines sur la ville en respectant un équilibre entre les trois piliers d'un développement durable (économique, social, et environnemental) sans privilégier l'un au détriment des autres ; (4) manipulation interactive des formes urbaines avec maintien des contraintes et préférences. Les deux dernières étapes sont au cœur de notre travail de thèse.



1. le concepteur fixe les contours de la ville, l'emplacement des centralités et l'emprise de chaque niveau d'intensité sur une grille régulière.
2. un système expert (basé sur des règles métier) calcule le nombre de formes urbaines[5] de chaque type (maison, bureau, commerce, industrie, ...) par niveau d'intensité en fonction d'un taux d'emploi et des capacités d'accueil du territoire ainsi délimité, de façon spécifique pour chaque pays.
3. les formes urbaines sont automatiquement réparties sur la grille régulière en respectant les contraintes de placement et en favorisant les préférences entre les formes urbaines, le tout traduisant un équilibre entre les différentes propriétés sociales, économiques et environnementales d'une ville durable.
4. les décideurs manipulent l'environnement urbain tandis que la cohérence des contraintes et des préférences est maintenue.

Nous cherchons à simuler une organisation spatiale optimisée d'un espace urbain durable dans le cadre d'une pré-programmation de façon à :

- respecter certaines contraintes structurelles ;
- tout en favorisant des préférences de placement liées aux formes urbaines.

[5]On parle ici du type dominante d'occupation du sol pour chaque cellule de la grille, à ne pas confondre avec la forme globale de l'espace urbain. C'est cette signification qui primera dans la suite de ce document.

4.2.1 Un carroyage symbolique régulier

Pour représenter la ville, nous utilisons une représentation symbolique prenant la forme d'un carroyage régulier où chaque carreau est desservi par quatre rues ou artères rectilignes (avenues, rues, dessertes, ...). Une fois définies, les voies de communication restent fixes sans entraîner de rétro-action avec les formes urbaines qui seront affectées aux carreaux. A notre niveau, nous nous intéressons plus particulièrement à la répartition des formes urbaines sur la ville (dans chaque carreau) et non aux formes géométriques qu'elle peut prendre. Une déformation à la marge de cette trame pourra être opérée par la suite (par exemple pour un rendu 3D plus réaliste) afin de l'adapter aux courbes du terrain ou aux traits de l'architecte et de l'urbaniste, sans pour autant modifier fondamentalement les résultats précédents.

Les données et ratios utilisés pour la conception de la plateforme, notamment le modèle urbain (figure 4.2, étape 2), s'appuient sur différentes réalisations urbaines afin de prendre en compte les contraintes techniques, commerciales, économiques, sociales et environnementales. Ce faisant, la plateforme SUSTAINS ne peut être exploitée en l'état que dans le cadre d'un développement urbain d'au moins cent cinquante hectares, plat et non urbanisé. Mais ces limitations ne concernent pas directement nos travaux qui peuvent être exploités avec ou en dehors de cette plateforme.

4.2.2 L'îlot comme unité de travail

Maintenant que nous avons identifié l'échelle globale d'un projet, il nous reste à définir la plus petite unité de travail pertinente pour notre travail. En France, pour cette taille de projet[6], une concertation préalable et une communication portant sur les enjeux et les objectifs de l'aménagement doivent être organisées et des séances de présentation au public sont généralement prévues pour évaluer différents scénarios[7]. Lors de cette phase pouvant coïncider avec l'étape de pré-programmation, les résultats présentés au grand public ne doivent pas apparaître trop réalistes pour favoriser le dialogue et renforcer l'idée que toutes les décisions ne sont pas arrêtées. Rappelons également que cette étape de PPU (Pré-Programmation Urbaine) doit proposer une vision à "gros grain" aux décideurs pour qu'ils ne s'embarrassent pas de détails superflus et qu'ils restent focalisés à un niveau suffisamment général dans le but de prendre des décisions stratégiques qui engageront le développement futur d'une ville entière ou d'un quartier sur le long terme. Dans ces circonstances, un découpage du territoire en cellules fixes de 80 m x 80 m correspondant à un îlot urbain standardisé semble suffisamment précis pour caractériser le niveau de granularité le plus fin d'une étude, sans devoir descendre à la parcelle ou au bâtiment. Ce choix renforce une représentation symbolique de la ville (pouvant néanmoins faire apparaître des bâtiments "simples") caractéristique d'une ébauche de projet pouvant encore évoluer.

4.2.3 Résolution optimale et résolution interactive

Intéressons-nous maintenant aux contextes de génération des différents scénarios. Les résultats que nous fournirons seront intégrés dans la plateforme selon deux protocoles :

- le premier protocole correspond à une recherche initiale de solutions pouvant être lancée sur plusieurs heures en ayant éventuellement recours à des ressources matérielles conséquentes ;
- le second protocole correspond à la modification interactive par les décideurs d'une solution initiale pré-calculée (i.e. issue du premier protocole).

Dans le premier protocole, nos algorithmes sont utilisés pour fournir une solution initiale (ou plusieurs propositions) dans un temps de résolution limité mais fixé à l'avance, l'échéance pouvant être de l'ordre de quelques heures à une nuit entière pour générer les différents scénarios de base. Dans le second protocole,

[6] La concertation préalable est obligatoire en France pour les aménagements d'espaces publics dépassant le seuil de 1,9 million d'euros d'investissement (cf. articles L 300-2 et R 300-1 du Code de l'Urbanisme).

[7] Un scénario correspond au résultat d'une simulation préalable retenue compte tenu de ses caractéristiques et pouvant représenter (ou être à la base) une solution potentielle. Différentes simulations permettent généralement d'obtenir différents scénarios présentant des caractéristiques différentes en fonction des choix opérés lors des processus de génération.

une solution initiale (basée sur un scénario) est projetée sur une grande table tactile et les décideurs interagissent avec la solution pour l'adapter à leurs visions ou décisions. Nos outils doivent alors tenir compte des modifications imposées par les utilisateurs tout en maintenant la cohérence des contraintes et des préférences dans un contexte ne laissant guère que quelques secondes tout au plus à la procédure de réparation pour opérer. Dans le premier cas, la priorité peut être donnée à la qualité des solutions produites alors que dans le second, c'est le temps de réponse qui devient prioritaire. Notre problème revient alors à positionner au mieux les formes urbaines issues du modèle urbain (figure 4.2, étape 2) sur chaque cellule libre de la grille, chaque cellule représentant un îlot urbain, ceci quel que soit le contexte de résolution (correspondant à nos deux protocoles). Sans en avoir la certitude à ce stade, nous faisons l'hypothèse qu'il sera possible de répondre aux exigences de ces deux contextes (pour la résolution initiale et les réparations interactives) grâce à une procédure d'optimisation unifiée capable de répondre de façon appropriée à chaque situation.

4.2.4 Positionnement de la PPU par rapport aux plateformes de conception urbaine existantes

Si il existe beaucoup de modélisations urbaines sur des sujets spécifiques, il n'y a pas précisément de modélisation relative à la pré-programmation urbaine. La PPU se positionne en amont des outils de simulation actuels. Elle vient compléter une chaîne méthodologique dans une tâche de programmation urbaine qui consiste à (i) modéliser une ville par ses caractéristiques et (ii) à simuler son évolution.

Parmi ces outils de simulation, on peut noter l'existence de la plateforme américaine URBANSIM [283] classée dans la catégorie des modèles d'urbanisation (Land-Use model). Ce logiciel peut fonctionner en interaction avec différents modèles externes de transport pour constituer un modèle complet de type LUTI (Land Use Transportation Integrated model).

En infographie, de multiples recherches se focalisent sur la création de villes nouvelles. Mais bien souvent, ces contributions s'attachent plus à la représentation esthétique des villes (en essayant de les imiter avec le plus de réalisme possible) qu'à leurs dimensions fonctionnelles. Néanmoins, les travaux récents de Vanegas et al. [273] tentent de combler ce vide en intégrant dans leurs outils les modèles d'évolution de la suite URBANSIM sans toutefois proposer une interaction utilisateur permettant de modifier en temps réel les caractéristiques de la ville et sans que cette plateforme soit conçue à l'intention expresse des urbanistes et des décideurs.

Finalement, il existe des logiciels élaborés pour les urbanistes (comme CommunityViz [158]) qui proposent des outils de conception automatisés et interactifs. Toutefois, ils se positionnent le plus souvent à une échelle très détaillée (parcelle, bâtiment) et la phase de conception pouvant être assimilée à la PPU reste pauvre en automatismes, privilégiant des tâches manuelles et répétitives assimilables à du dessin.

4.3 Problématique

Nous évaluons mieux après ces quelques pages l'étendue et la complexité du sujet traité dans cette thèse. Pour l'aborder de façon méthodique, nous proposons de décomposer notre problématique en trois grandes sections complémentaires :

- proposer un modèle de pré-programmation urbaine ;
- rechercher une bonne organisation spatiale de la ville à partir du modèle de PPU dans un contexte batch ou interactif ;
- imaginer une interaction avec l'utilisateur capable de maintenir la cohérence globale d'une solution.

4.3.1 Proposer un modèle de pré-programmation urbaine

Le modèle proposé doit nécessiter très peu de données en entrée, ces données pouvant être limitées aux éléments suivants :

- les contours du territoire à aménager ;
- l'emprise de chaque intensité ;
- le nombre et la position de chaque centralité ;
- un carroyage régulier définissant les axes de transport principaux et secondaires (avenues, rues et dessertes) ;
- les zones non constructibles (rivières, forêts à préserver, ...) ;
- le nombre de chaque forme urbaine à répartir sur la ville, ces formes urbaines étant ventilées par niveau d'intensité.

Le carroyage permet de découper le territoire en cellules régulières où chaque cellule représente un îlot urbain. Une cellule peut être fixe ou libre. Pour une cellule fixe, la forme urbaine est imposée (rivière, forêt, ...) alors que pour une cellule libre, aucune forme urbaine ne lui est affectée. Tout l'enjeu consiste à associer à chaque cellule libre la meilleure forme urbaine possible en fonction des formes urbaines à répartir sur la ville. La partie relative à la modélisation est très délicate compte tenu du nombre important de notions à traiter et de la difficulté à les représenter. De plus, il faut appréhender le problème dans sa globalité (approche systémique), avec par exemple une action simultanée touchant à la localisation des diverses ressources territoriales, la densification ou encore des agencements favorables aux modes de transport doux ou collectifs.

Par rapport à cette section, les questions suivantes se posent alors :

- une phase d'analyse approfondie avec des experts de l'aménagement urbain permettra-t-elle d'identifier les propriétés d'un modèle de PPU pertinent avec si peu de données disponibles en entrée ?
- est-il possible de transposer ou d'adapter un modèle américain aux autres continents ?
- notre modèle sera-t-il circonscrit à la création d'une ville nouvelle ou pourrait-il être utilisé pour l'extension d'une ville existante ou la réhabilitation d'un quartier (friche industrielle, ...) ?
- comment définir et représenter les règles de placement des formes urbaines pour satisfaire aux critères d'un développement durable ?
- une propriété donnée du problème doit-elle être traduite en contrainte ou en préférence ?
- le modèle proposé sera-t-il suffisamment simple pour être compréhensible par des utilisateurs non experts (décideurs), critère essentiel pour son adoption ?
- quel formalisme doit-on adopter pour représenter notre modèle ?

4.3.2 Rechercher une bonne organisation spatiale de la ville à partir du modèle de PPU dans un contexte batch ou interactif

L'objectif consiste à générer (contexte batch) ou à maintenir (contexte interactif) une ville durable optimisée et réaliste. Pour cela, nous allons exploiter notre modèle de PPU pour répartir au mieux et de façon automatique les formes urbaines (calculées par le modèle urbain) sur chaque cellule libre du carroyage. On entend par libre une cellule dont la forme urbaine n'a été imposée ni par le modèle urbain, ni par l'utilisateur.

Reste alors à répondre aux difficultés suivantes :

- y a-t-il toujours une solution au problème posé ?
- inversement, si il existe plusieurs solutions au problème, alors ces solutions sont-elles nombreuses et pouvons nous trouver la meilleure solution ?
- existe-t-il des méthodes capables de résoudre efficacement notre problème et seront-elles en mesure de passer à l'échelle pour des villes réelles de différentes tailles ou de différentes natures ?
- est-ce que la qualité des résultats obtenus peut être garantie ?
- face à une instance d'un problème particulier, est-t-il possible de prédire les temps ou la puissance de calcul nécessaire ?
- les méthodes de résolution retenues pourront-elles s'adapter à la fois au contexte batch et au contexte interactif ou bien faut-il prévoir une méthode dédiée à chaque contexte ?
- faudra-t-il dégrader le modèle pour le rendre compatible avec la partie interactive ?

- ces méthodes pourront-elles être intégrées dans un solveur unifié ?
- si la taille du problème devient trop importante, la résolution pourra-t-elle encore se faire sur un ordinateur classique ou bien faut-il prévoir une grille de calcul pour pouvoir augmenter la taille des problèmes à traiter tout en conservant des temps de calcul raisonnables ?

4.3.3 Imaginer une interaction avec l'utilisateur capable de maintenir la cohérence globale d'une solution

Bien qu'elle soit abordée en dernier, cette section occupe une place essentielle dans notre problématique de par son aspect « aide à la décision » en lien directe avec les utilisateurs. Dans cette partie, les décideurs doivent interagir directement avec un scénario et le système informatique sous-jacent doit être en mesure de répondre en temps réel aux actions opérées pour maintenir de façon automatique la cohérence globale de la solution en cours.

Bien qu'il faille représenter des scénarios réalistes sur une table tactile de grande surface avec des quartiers, des îlots, des parcelles et des bâtiments de différentes natures (maison d'habitation, immeuble collectif, bureau, usine, etc.) en y intégrant les rues, espaces verts, espaces naturels et équipements publics, nous limiterons nos travaux à une représentation symbolique (et non géométrique) de la ville discrétisée par une grille uniforme où chaque cellule représente un îlot urbain associé à une forme urbaine dominante.

Plus précisément, nous devrons répondre aux interrogations suivantes :

- quelles seront les manipulations proposées à l'utilisateur, seront-t-elles limitées ou étendues, auront-elles une portée locale ou globale ?
- faut-il prévoir plusieurs échelles de modification ou des niveaux hiérarchiques pour appréhender le problème interactif ?
- des contraintes peuvent-elles être violées suite à une opération, et si oui, quelles sont ces contraintes ?
- quelle forme doit prendre le processus de résolution : doit-il rester actif pendant toute la phase d'interaction ou peut-on le relancer en réponse à un événement particulier ? Quels peuvent-être ces différents événements et comment les propager à la procédure de résolution ?
- peut-on gérer la réparation d'une solution comme une résolution initiale et faudra-t-il incorporer des données supplémentaires dans le modèle pour gérer l'interaction ?
- avec l'utilisation de tables tactiles multi-touch, faut-il prévoir plusieurs utilisateurs opérant des modifications simultanées sur le même scénario ?
- en retour à chaque action, comment informer les décideurs des conséquences de leurs choix, à partir de quels indicateurs ?
- comment représenter ces indicateurs et pourront-ils être calculés en temps réel ?
- sur quels critères pourra-t-on évaluer la pertinence du processus interactif ?

4.4 Méthode générale de travail

Pour répondre à toutes les questions qui se posent, nous allons étudier des projets qui se sont confrontés à des problématiques voisines pour enrichir nos connaissances et trouver la bonne façon de les aborder. Nous nous appuierons également sur des professionnels du secteur, en particulier les urbanistes, architectes et concepteurs d'IHM[8] impliqués dans le projet SUSTAINS, afin d'appréhender toutes les notions importantes à intégrer. Grâce à des confrontations directes et régulières, des réponses concrètes pourront être apportées au fil des rencontres. Plus tard, le point de vue des relecteurs de nos articles de recherche constituera probablement une source utile tout comme les présentations scientifiques et les discutions associées.

Nous considérons que la réalisation de prototypes constitue un moyen objectif et efficace pour évaluer différentes classes de méthodes sur nos problèmes, les comparer et savoir si elles répondent bien aux objec-

[8] Interface Homme Machine

tifs fixés. Bien que la partie IHM ne soit pas directement de notre ressort, nous allons tout de même proposer une procédure de saisie et de rendu graphique très simple gérant l'interaction avec les utilisateurs pour appréhender l'ensemble des difficultés relatives aux modifications temps réel d'une solution et proposer en retour des moyens pertinents d'informer les utilisateurs des conséquences de leurs choix.

Un aspect essentiel concerne la validation des résultats. Pour cela, nous travaillerons sur différentes instances de nos problèmes : dans un premier temps avec des instances « jouets » plus ou moins grosses pour les mises au point et l'évaluation des performances au sens informatique (temps de calcul, qualité des résultats), et dans un second temps, avec des instances réelles provenant de Marne-la-vallée. A chaque fois, les résultats seront soumis à des professionnels de l'aménagement urbain pour être validés.

Pour ce qui concerne la cohabitation de nos recherches avec le projet SUSTAINS, elle constitue une réelle opportunité mais revêt également bien des difficultés. Par exemple, cela nécessite de synchroniser l'avancement de la thèse avec le calendrier du projet en ne perdant pas de vue les objectifs communs et les échéances qui ont été définis, ce qui peut favoriser une certaine rigidité. De plus, les décisions sur des sujets aussi vastes que l'urbanisme peuvent dans certains cas déchaîner les passions ou donner lieu à des discutions sans fin. D'un point de vue opérationnel et en l'absence d'une plateforme technique commune, les différents modules doivent communiquer par le biais de services distants répartis sur les sites de chaque partenaire en respectant des normes précises pour permettre aux différents services de travailler sur des données partagées. Et compte tenu des accords de consortium, certains documents ou algorithmes peuvent rester confidentiels de part et d'autre. Sous oublier que l'interdisciplinarité nécessite l'adoption d'un vocabulaire commun pour réussir à se comprendre, une ouverture d'esprit sur des problématiques qui ne sont pas directement en lien avec ses propres thèmes de recherche ou encore une adaptation permanente à des niveaux de réflexion qui peuvent être très généraux ou très précis. Pour faciliter les choses, nos réalisations doivent pouvoir fonctionner à la fois de façon intégrée avec la plateforme SUSTAINS, et de façon complètement autonome pour ne pas être confronté à ses limitations (indisponibilité d'un module, bugs, retards, expérimentations sur des échelles différentes, ...).

En tout état de cause, nous faisons ici l'hypothèse forte que les avantages d'une collaboration pluridisciplinaire surclasseront largement tous les inconvénients, et qu'au final la plupart des difficultés se révéleront être très probablement des opportunités (rigueur, consensus, organisation, partage du savoir, ...).

4.5 Conclusion

À partir d'une synthèse des éléments contextuels qui portent d'une part sur l'apport de l'urbanisme au développement durable et d'autre part sur le projet de recherche SUSTAINS auquel nous participons, un cadre méthodologique suffisamment précis a pu être identifié. Ces éléments nous ont permis de dégager une problématique détaillée relative à cette thèse que nous avons décliné en trois grands aspects portant sur la conception d'un modèle urbain, les possibilités de résolution correspondantes et la proposition d'une interaction avec les utilisateurs.

Pour répondre aux nombreuses questions posées et aux difficultés scientifiques qu'elles soulèvent, nous allons commercer par dresser un état de l'art complet qui porte sur trois sujets complémentaires :

- les problèmes d'optimisation combinatoire sous contraintes et les méthodes de résolutions associées ;
- les systèmes interactifs d'aide à la décision ;
- la modélisation urbaine et les problèmes de positionnement.

II

État de l'art

5

Problèmes d'optimisation combinatoire sous contraintes

Comment trouver le plus court chemin passant par vingt-cinq villes données sans devoir examiner un à un les milliards de trajets possibles ? Ce sont de tels problèmes qu'informaticiens et mathématiciens étudient depuis plusieurs décennies pour trouver des méthodes de résolution efficaces capables de fournir des solutions avec un temps de réponse acceptable. Trouver une bonne solution consiste parfois à faire gagner du temps ou de l'argent, mais avec les avancées scientifiques et techniques, nous devons faire face à de nouveaux problèmes. C'est le cas par exemple de l'optimisation combinatoire appliquée à la génomique où l'on s'efforce de décrypter l'information génétique d'un organisme vivant.

Les enjeux liés à la résolution des problèmes combinatoires sont donc essentiels, que cela soit sur le plan scientifique pour produire du savoir algorithmique, ou sur le plan pratique pour améliorer la performance de systèmes de plus en plus complexes (systèmes mécaniques, traitement d'images, électronique, ...). C'est ainsi que l'optimisation combinatoire occupe une place très importante en recherche opérationnelle et en informatique. De nombreuses applications peuvent être modélisées sous la forme d'un problème d'optimisation combinatoire telles que le problème du voyageur de commerce, l'ordonnancement de tâches, le problème de coloration de graphes, etc.

Après l'identification des caractéristiques essentielles relatives aux problèmes combinatoires, nous ferons l'inventaire des principales méthodes capables d'adresser ces problèmes, ces méthodes pouvant être classées dans la catégorie des méthodes complètes ou incomplètes. Nous nous intéressons plus précisément à deux méthodes de résolution efficaces applicables à ces problèmes : les méthodes complètes[1] de programmation par contraintes (PPC) et les techniques de recherche locale (RL) qui sont des méthodes incomplètes[2].

Il existe bien d'autres techniques de résolution, comme les algorithmes génétiques [133], de colonies de fourmis [78], l'optimisation par essaims particulaires [150] ou encore les systèmes multi-agents [81, 92]. Néanmoins, le sujet est si vaste qu'il est impossible ici d'aborder toutes les techniques de résolution en détail.

Les problèmes d'optimisation étant de plus en plus complexes et les besoins en ressources ne cessant d'augmenter, les métaheuristiques réputées pouvoir attaquer des problèmes très difficiles peuvent à leur tour être limitées face à ces nouvelles exigences. Nous verrons alors comment le parallélisme permet d'étendre les capacités de ces techniques sans remettre en cause leurs fondements.

[1] Peuvent prouver l'insatisfiabilité ou l'optimalité

[2] A l'inverse, elles ne sont pas en mesure de prouver l'insatisfiabilité ou l'optimalité

Ce chapitre constitue une synthèse des travaux et documents de référence suivants : [16, 251, 226, 14, 76, 30, 47, 115, 44, 12].

5.1 Caractéristiques des problèmes combinatoires

Les problèmes de nature combinatoire possèdent un nombre fini de configurations possibles, mais ce nombre est très grand, de sorte qu'une résolution par une simple énumération n'est pas concevable dans la pratique. Ces problèmes sont souvent faciles à définir mais habituellement difficiles à résoudre [127].

5.1.1 Problème d'optimisation combinatoire

Explosion combinatoire Rechercher le chemin le plus court ou le plus rapide entre deux points est un problème assez simple à résoudre. Mais quand le nombre d'étapes croît, le nombre de possibilités augmente au-delà de ce que l'on peut énumérer de tête, et même au delà de ce que peuvent examiner les ordinateurs les plus puissants en un temps raisonnable : il s'agit de l'*explosion combinatoire*.

Il existe bien souvent un nombre exponentiel de solutions à un problème en fonction de sa taille. Le problème du voyageur de commerce (Traveling Salesman Problem, TSP) est un problème caractéristique d'optimisation combinatoire. Pour ce problème, il s'agit de trouver un chemin qui passe une fois et une seule par chaque ville en minimisant la distance totale parcourue pour revenir au point de départ. Si n représente le nombre de villes et $\mathcal{S}$ l'ensemble des partitions d'un ensemble à n éléments, alors $|\mathcal{S}| = \frac{(n-1)!}{2}$ correspond au nombre de chemins possibles pour un problème symétrique.

Une résolution du problème avec une énumération exhaustive consiste à générer tous les trajets possibles, à calculer leurs distances et à choisir le trajet ayant la distance minimale. La table 5.1 illustre le phénomène d'explosion combinatoire pour le problème du voyageur de commerce avec un nombre croissant de villes à parcourir.

TABLE 5.1 – Problème du voyageur de commerce : temps de calcul nécessaire pour trouver le trajet le plus court en réalisant une énumération exhaustive de tous les trajets possibles en fonction du nombre de villes (un trajet pouvant être évalué en une microseconde). D'après [24].

n	$\lvert\mathcal{S}\rvert$	Temps de calcul
5	12	12 microsecondes
10	181 440	0,18 seconde
15	43 milliards	12 heures
20	6 E+16	19 siècles
25	3 E+23	9,8 milliards d'années

Pour repousser les limites et résoudre des problèmes d'optimisation combinatoire de plus grande taille en un temps raisonnable, on peut certes compter sur l'augmentation des capacités matérielles mais bien plus encore sur l'innovation algorithmique [24].

Definition 1 (Problème d'optimisation combinatoire)**.** Un problème d'optimisation combinatoire est généralement caractérisé par un ensemble fini de solutions admissibles $\mathcal{S}$ et une fonction objectif $f : \mathcal{S} \rightarrow \mathbb{R}$ qui associe une valeur à chaque solution admissible. La résolution du problème d'optimisation consiste à déterminer la ou les solutions de $\mathcal{S}$ minimisant ou maximisant f.

Lorsque la fonction f représente un coût ou une perte, on parle d'un problème de minimisation. Lorsque la fonction f représente un gain, on parle d'un problème de maximisation.

5.1.2 Complexité

Un algorithme nécessite deux ressources importantes pour résoudre un problème : du temps et de l'espace. La complexité en temps d'un algorithme correspond à une estimation du nombre d'instructions (ou d'étapes) à exécuter pour résoudre les instances d'un problème de taille n, cette estimation étant donnée en ordre de grandeur dans le pire cas, c'est à dire pour l'instance la plus difficile à résoudre. La complexité d'un problème est équivalente à la complexité du meilleur algorithme (au sens du plus rapide) pouvant résoudre ce problème. Comme les problèmes combinatoires ne possèdent pas tous la même difficulté, on utilise une classification définie par la théorie de la complexité pour cerner le niveau de difficulté lié à leur résolution [214]. Cette classification prend ses sources en 1971, lorsque Stephen COOK et Leonid LEVIN ont formulé quasiment au même moment deux classes importantes de problèmes de décision :

- la classe $\mathcal{P}$: correspond à l'ensemble des problèmes de décision algorithmiques pouvant être résolus en un temps polynomial (problèmes relativement faciles à résoudre) ;
- la classe $\mathcal{NP}$: correspond à l'ensemble des problèmes de décision algorithmiques tels que, si une solution possible est donnée, il est possible de vérifier cette solution en un temps polynomial (problèmes faciles à vérifier mais difficiles à résoudre).

Un problème de décision correspond à une question mathématique qui a toujours une solution pouvant s'exprimer par "oui" ou par "non". Un problème d'optimisation peut toujours être réduit à un problème de décision.

La classe de complexité $\mathcal{P}$ contient l'ensemble de tous les problèmes de décision pouvant être résolus par un algorithme de complexité polynomiale[3]. Cette classe regroupe les problèmes qui peuvent être résolus efficacement. La classe de complexité $\mathcal{NP}$ rassemble quant à elle les problèmes de décision pouvant être résolus par un algorithme de complexité polynomiale pour une machine non déterministe, c'est à dire capable d'évaluer en parallèle un nombre fini d'alternatives. La classe $\mathcal{NP}$ englobe ainsi la classe $\mathcal{P}$. La résolution des problèmes $\mathcal{NP}$ peut nécessiter l'examen d'un grand nombre de cas, éventuellement exponentiel[4], mais chaque cas doit pouvoir être examiné en un temps polynomial. Pour les problèmes les plus difficiles de $\mathcal{NP}$, on ne trouve jamais d'algorithme polynomial pour résoudre chaque cas avec une machine déterministe. Ces derniers problèmes définissent la classe des problèmes $\mathcal{NP}$*-complet*.

Definition 2 ($\mathcal{NP}$*-complet*)**.** Un problème est dit $\mathcal{NP}$*-complet* si :

- il est dans $\mathcal{NP}$;
- et il est au moins aussi difficile que tout problème de $\mathcal{NP}$.

Si on ne tient pas compte de la clause d'appartenance à la classe $\mathcal{NP}$, on trouve la classe des problèmes $\mathcal{NP}$*-difficile* qui fait référence aux problèmes d'optimisation dont les problèmes de décision associés sont $\mathcal{NP}$*-complet*. Ces problèmes $\mathcal{NP}$*-difficile* sont au moins aussi difficiles à résoudre que n'importe quel problème de $\mathcal{NP}$ [252]. Pour ces problèmes $\mathcal{NP}$*-difficile*, on présume qu'il n'existe pas d'algorithme polynomial pour les résoudre à l'optimalité, c'est à dire un algorithme dont le temps de calcul soit proportionnel à n^i, n étant le nombre de variables inconnues du problème et i une constante entière.

Definition 3 ($\mathcal{NP}$*-difficile*)**.** Un problème est dit $\mathcal{NP}$*-difficile* si il est au moins aussi difficile que tout problème de $\mathcal{NP}$. La catégorie $\mathcal{NP}$*-difficile* inclut également les problèmes qui ne sont pas des problèmes de décision.

Beaucoup de problèmes académiques populaires sont $\mathcal{NP}$*-difficile*, parmi eux, nous pouvons citer :

- les problèmes d'affectation-localisation : problème d'affectation quadratique (Quadratic Assignment Problem ou QAP), problème d'affectation généralisé (Generalized Assignment Problem ou GAP), problème de localisation d'équipements (Facility Location Problem), problème de P-médian ;

[3] Complexité polynomiale notée $O(n^i)$: quand n double, le temps d'exécution est multiplié par 2^i ; n étant le nombre de variables inconnues du problème et i une constante entière.

[4] Complexité exponentielle notée $O(i^n)$: quand n double, le temps d'exécution est élevé à la puissance 2 ; n étant le nombre de variables inconnues du problème et i une constante entière.

- les problèmes de regroupement : Partitionnement de données (Data Clustering), Coloration de graphe (Graph Coloring)
- le problème de couverture par ensembles (Set Cover Problem ou SCP).

En fait, une majorité des problèmes de la vie réelle [104, 266] appartiennent à la classe des problèmes *$\mathcal{NP}$-difficile* et ne possèdent pas de solution algorithmique efficace valable pour les résoudre à l'optimalité lorsque la taille du problème augmente.

5.1.3 Problèmes de satisfaction de contraintes

Un très grand nombre de problèmes combinatoires appartient à la famille des *problèmes de satisfaction de contraintes* (Constraint Satisfaction Problem ou CSP) : configuration, planification, ordonnancement, affectation de ressources, etc. Étant donné un ensemble de contraintes, portant chacune sur un sous-ensemble donné de variables auxquelles on cherche à affecter une valeur, la question centrale est de savoir s'il existe une possibilité de satisfaire simultanément toutes ces contraintes et, le cas échéant, quelles valeurs peut-on affecter à chaque variable pour satisfaire toutes les contraintes [27]. Ces problèmes partagent une description commune, basée sur un formalisme très simple, qui autorise en général une modélisation claire et intuitive.

Definition 4 (CSP)**.** Un problème de satisfaction de contraintes (CSP) est défini par un triplet (X, D, C) tel que :

- $X = \{x_1, \cdots, x_n\}$ est l'ensemble des variables (les inconnues) du problème,
- $D = \{D(x_1), \cdots, D(x_n)\}$ est l'ensemble des domaines et $D(x_i)$ correspond à l'ensemble fini des valeurs possibles pour la variable x_i,
- $C = \{C_1, C_2, ..., C_k\}$ représente l'ensemble des contraintes et C_j est une relation entre certaines variables de X. Une contrainte définit les combinaisons de valeurs des variables autorisées. C'est une relation logique (une propriété devant être vérifiée) entre différentes variables. Ainsi, une contrainte restreint les valeurs que peuvent prendre simultanément les variables.

Definition 5 (Affectation)**.** On appelle *affectation* (ou *configuration*) le fait d'instancier certaines variables à des valeurs prises dans leurs domaines respectifs. Une affectation est dite *totale* si elle instancie toutes les variables et *partielle* si elle n'en instancie qu'une partie. Une affectation (totale ou partielle) est *consistante* si elle ne viole aucune contrainte, et *inconsistante* si elle viole une ou plusieurs contraintes.

Definition 6 (Espace de recherche)**.** L'espace de recherche d'un CSP $P = (X, D, C)$ est l'ensemble des configurations possibles noté S, tel que :

$$S = D(x_1) \times D(x_2) \times \cdots \times D(x_n)$$

L'espace de recherche est égal au produit cartésien de l'ensemble des domaines des variables et il croît de façon exponentielle avec le nombre de variables.

Definition 7 (Solution)**.** Une solution d'un CSP $P = (X, D, C)$ est une affectation totale $s \in S$ qui satisfait toutes les contraintes (consistante). $\mathcal{S}$ correspond à l'ensemble des solutions de P tel que :

$$\mathcal{S} = \{s \in S \mid \forall c \in C, s \text{ satisfait } c\}$$

Résoudre un CSP consiste alors à trouver une solution réalisable, un nombre donné de solutions réalisables ou toutes les solutions réalisables.

5.1.4 Problèmes d'optimisation sous contraintes

Beaucoup de problèmes combinatoires peuvent également s'exprimer comme des *problèmes d'optimisation sous contraintes* (Constraint Satisfaction and Optimization Problem ou CSOP).

Un problème d'optimisation sous contraintes est un problème pour lequel on cherche parmi l'ensemble de toutes les solutions réalisables la meilleure solution selon une fonction qui définit un objectif donné. La fonction *objectif* a pour rôle d'évaluer la qualité d'une solution tandis que les contraintes permettent d'éliminer les configurations qui ne sont pas des solutions. Le problème est alors double : le premier consiste à trouver l'ensemble de toutes les solutions réalisables et le deuxième à trouver dans cet ensemble la meilleure solution qui minimise ou maximise la fonction objectif. On parle d'un *problème de minimisation* quand la qualité d'une solution est donnée par une fonction objectif à minimiser et d'un *problème de maximisation* quand la qualité d'une solution est donnée par une fonction objectif à maximiser. L'étude d'un problème d'optimisation révèle habituellement des contraintes impératives (structurelles ou dures) et des contraintes indicatives (préférences ou molles) [46]. Les solutions retenues doivent respecter les contraintes impératives qui limitent l'espace de recherche tandis que les contraintes indicatives doivent être respectées autant que possible, le niveau de satisfaction de ces dernières devant être encodé dans la fonction objectif.

Definition 8 (CSOP)**.** Un problème d'optimisation sous contraintes (CSOP) est défini par le quadruplet (X, D, C, f) avec :

- $X = \{x_1, \cdots, x_n\}$ est l'ensemble des variables du problème,
- $D = \{D(x_1), \cdots, D(x_n)\}$ est l'ensemble des domaines,
- $C = \{C_1, C_2, ..., C_k\}$ représente l'ensemble des contraintes,
- f est une fonction objectif définie sur un sous-ensemble de X (à minimiser ou à maximiser).

La fonction f permet de définir une relation d'ordre total entre n'importe quelle paire de solutions dans $\mathcal{S}$. Nous avons vu à la section 5.1.2 qu'on pouvait associer un problème de décision à chaque problème d'optimisation, le but étant de déterminer s'il existe une solution pour laquelle la fonction objectif soit supérieure (respectivement inférieure) ou égale à une valeur donnée. Ainsi, la complexité d'un problème d'optimisation est liée à celle du problème de décision qui lui est associé. En particulier, si le problème de décision est $\mathcal{NP}$*-complet*, alors le problème d'optimisation est $\mathcal{NP}$*-difficile* [252].

5.1.4.1 Multiobjectif versus multicritère

Pour certains problèmes combinatoires, plusieurs objectifs contradictoires doivent être optimisés en même temps. Par exemple, lors de la préparation d'un voyage, il peut être demandé de choisir entre différents modes de déplacement (avion, train, bateau, voiture, etc.) de façon à minimiser le coût de transport tout en minimisant le temps du trajet.

On recense essentiellement deux approches associées à ces problèmes [51] : les techniques de génération (que nous appellerons *multiobjectif*) et les techniques à base de préférences (que nous nommerons *multicritère*). Les techniques basées sur les préférences consistent à synthétiser les informations relatives à différents points de vue ou aspects. Il s'agit de ramener un problème multiobjectif à un problème simple objectif. Elles utilisent une méthode pour classer les objectifs et ainsi trouver une solution qui optimise le classement. Ce classement peut être réalisé par une variété de moyens allant d'une simple somme pondérée des objectifs à une fonction d'utilité complexe [33, 3]. Les techniques de génération quant à elles recherchent dans un ensemble très grand de solutions celles qui satisfont au mieux l'ensemble des objectifs. Pour cela, elles identifient les configurations optimales de Pareto où chaque configuration est telle qu'il est impossible d'améliorer le score de l'une sans diminuer le score de l'autre.

Dans un contexte d'aide à la décision, les problèmes contiennent souvent une multitude d'aspects simultanés à prendre en considération pour la prise de décision. Pour S. BEN-MENA [25], l'approche multicritère,

capable d'intégrer tout type de critère[5], permet alors de se diriger vers un compromis judicieux plutôt qu'un optimum qui n'a pas toujours de sens. Une technique classique consiste alors à encoder chaque aspect du problème dans la fonction objectif, en attribuant à chaque critère un éventuel poids pour lui accorder plus ou moins d'importance par rapport aux autres critères [234]. Bien que cette approche du « *critère unique de synthèse* » soit soumise à une compensation possible entre critères ou à une forte sensibilité aux changements d'échelle, les méthodes d'agrégation complète peuvent s'avérer intéressantes ou tout simplement les seules utilisables [240].

5.1.5 Problèmes d'optimisation dynamique sous contraintes

Les problèmes d'optimisation dynamique sont présents dans beaucoup d'applications réelles et représentent à ce titre un défi important. Ils sont caractérisés par des éléments en entrée du problème qui changent au fil du temps [266]. Par exemple, ces changements peuvent être liés à des machines qui tombent en panne, à une qualité variable de matières premières ou encore à l'introduction de nouvelles tâches à planifier. Cela peut concerner des problèmes classiques comme le problème de tournées de véhicules (Vehicle Routing Problem ou VRP), les problèmes d'ordonnancement de tâches ou des problèmes plus spécifiques comme le contrôle de réactions chimiques liées à des conditions de température et de pression [168].

Bien que le formalisme et les algorithmes CSP permettent de représenter et de traiter de nombreux problèmes d'intelligence artificielle et de recherche opérationnelle, le cadre CSP montre des limitations face à des problèmes dynamiques du monde réel dans lesquels la définition du problème est modifiée incrémentalement. La notion de CSP dynamique (Problème de Satisfaction de Contraintes dynamique ou Dynamic Constraint Satisfaction Problem) a été introduite par [68] pour représenter de telles situations. Un DCSP est une séquence de CSP dont chaque élément diffère du précédent par l'ajout et/ou le retrait de certaines contraintes.

Concernant la fonction objectif associée au problème d'optimisation, et bien qu'elle soit déterministe, elle change également au fil du temps au gré des modifications qui interviennent en entrée du problème [144] : $f_{dynamic}(s) = f_t(s)$ où t représente le temps au cours duquel la fonction objectif est évaluée. Et lorsque la fonction objectif change, il y a de fortes chances pour que la solution optimale du problème change également.

Compte tenu de ces éléments, les principales difficultés à surmonter pour un problème d'optimisation dynamique sont [34, 198] :

- détecter le changement dans l'environnement lorsqu'il se produit ;
- répondre à ce changement pour suivre la nouvelle solution optimale.

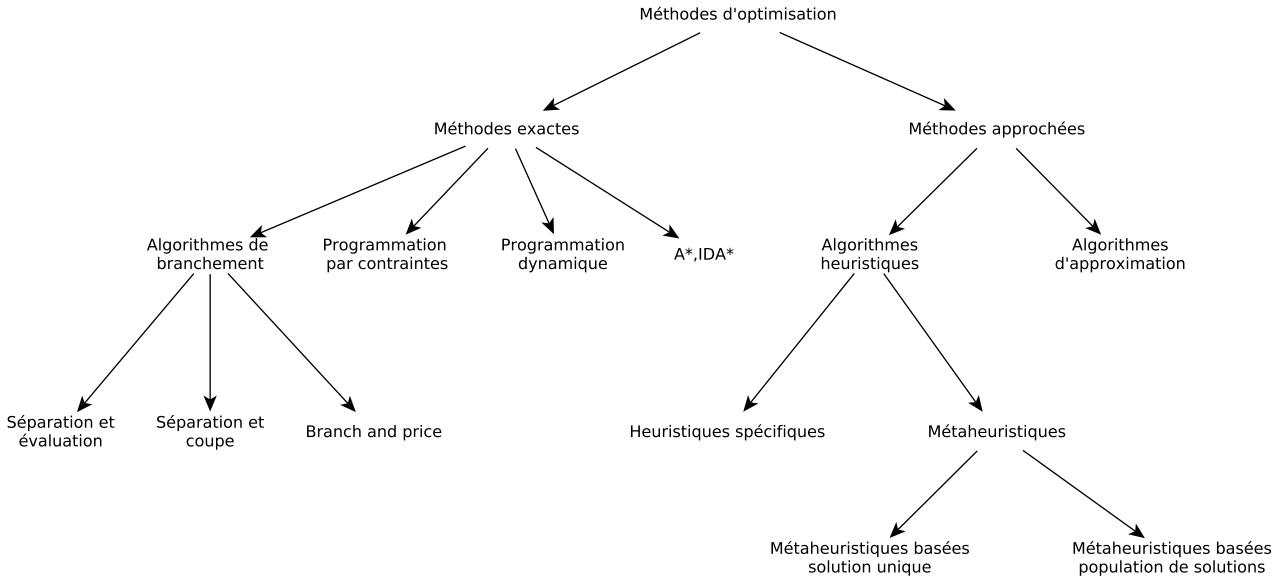
Le processus de recherche doit alors s'adapter rapidement au changement de la fonction objectif. Le but n'est plus seulement de trouver l'optimum global, mais de le suivre aussi fidèlement que possible dans le temps [168]. Le principal défi consiste à réutiliser l'information des recherches précédentes pour s'adapter au changement du problème au lieu de résoudre à nouveau le problème.

5.2 Résolution des problèmes d'optimisation combinatoire

Les problèmes d'optimisation combinatoire constituent une catégorie de problèmes très difficiles à résoudre [215]. La résolution d'un problème d'optimisation combinatoire consiste à explorer un espace de recherche afin de minimiser ou maximiser une fonction objectif pour trouver la meilleure solution, définie comme la solution globalement optimale ou optimum global. Cette résolution est habituellement délicate puisque le nombre de solutions réalisables croît généralement avec la taille du problème, rendant un parcours exhaustif impraticable en pratique.

[5] Un critère permet de mesurer les préférences du décideur vis-à-vis d'une action. Par exemple, pour l'achat d'une voiture : coût à l'achat, confort, sécurité, économie à l'usage ; ou pour le choix du meilleur emplacement pour la construction d'une station d'épuration : nuisances olfactives, coût d'acheminement de l'eau, ...

FIGURE 5.1 – Méthodes classiques d'optimisation. D'après le livre [266].



La complexité en taille ou en structure de l'espace de recherche d'une part et de la fonction objectif d'autre part peut conduire à utiliser des méthodes de résolution différentes en fonction du problème ou de l'instance à traiter. Intuitivement, on pourrait penser que la difficulté d'une instance d'un problème est proportionnelle au nombre de variables, à la taille du domaine par variable et au nombre de contraintes. Dans les faits, il est le plus souvent impossible de prévoir avec certitude l'efficacité d'une méthode donnée pour un problème particulier. Le théorème "*no free lunch*" ("*pas de dîner gratuit, NFL*") [288] nous indique, sous certaines hypothèses, qu'une méthode de résolution ne peut prétendre être la plus efficace sur tous les problèmes.

Les méthodes de résolution liées aux problèmes d'optimisation peuvent être regroupées en deux grandes familles (figure 5.1) :

- les méthodes dites *complètes* (ou exactes) qui obtiennent les solutions optimales et garantissent les résultats ;
- les méthodes dites *incomplètes* (ou approchées) qui trouvent rapidement des solutions de bonne qualité sans garantir l'optimalité.

L'optimisation combinatoire couvre un large éventail de techniques et fait toujours l'objet de recherches intensives sur de nombreux domaines d'application (finance, transport, surveillance, biologie, ...) de sorte qu'il est impossible de toutes les aborder ici en détail. Nous nous limitons dans un premier temps à un panorama non exhaustif de chaque famille pour avoir une idée très générale des méthodes existantes.

5.2.1 Panorama des méthodes complètes

Les algorithmes complets (ou exacts) évaluent l'espace de recherche dans sa totalité en réalisant une énumération intelligente. Ils donnent la garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et permettent de prouver son optimalité [222].

5.2.1.1 La programmation linéaire

La *programmation linéaire* propose un cadre de modélisation mathématique permettant de résoudre des problèmes d'optimisation avec des algorithmes efficaces comme la méthode du *simplexe* [62] ou les méthodes de *points intérieurs* [148]. On parle de programme linéaire en nombres entiers (PLNE) lorsque les

valeurs des variables du programme linéaire doivent être des nombres entiers et de programme linéaire mixte (PLM) lorsque certaines variables doivent être entières et d'autres pas. *Relaxer* un PLNE ou un PLM consiste en général à supprimer certaines de ses contraintes ou à les remplacer par des contraintes plus faibles. Le problème résultant est souvent plus simple à résoudre que le problème original et sa solution optimale peut être soit réalisable pour le problème original, donc optimale pour ce dernier, soit non réalisable et avoir un coût proche de l'optimum du problème original.

5.2.1.2 L'algorithme de séparation et évaluation

L'algorithme de *séparation et évaluation* (Branch and Bound) [163] utilise une méthode arborescente de recherche basée sur l'idée d'énumérer implicitement les solutions possibles d'un problème de PLNE pour trouver la meilleure solution sans passer par une énumération explicite trop gourmande en temps. Cet algorithme est composé de trois éléments principaux :

- la séparation : consiste à diviser le problème en sous-problèmes où chaque sous-problème contient un ensemble de solutions réalisables tel que l'union de leurs espaces de solutions forme l'espace des solutions du problème père. Cela revient à construire un arbre (arbre de recherche ou arbre de décision) permettant d'énumérer toutes les solutions. Ainsi, en résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial.
- l'évaluation : permet de réduire l'espace de recherche en éliminant les sous-ensembles qui ne contiennent pas la solution optimale. Le principe consiste à mémoriser la solution de plus bas coût (pour un problème de minimisation) rencontrée pendant l'exploration et à comparer le coût de chaque nœud parcouru avec celui de la meilleure solution. Si le coût du nœud considéré est supérieur au meilleur coût trouvé, on peut arrêter l'exploration de la branche car toutes les solutions de cette branche seront nécessairement moins bonnes.
- une stratégie de parcours : en largeur, en profondeur ou le meilleur d'abord.

5.2.1.3 Les méthodes de coupes

Les *méthodes de coupes* ou *plans sécants* introduites par GOMORY [119] permettent de simplifier un PLNE en relaxant certaines de ses contraintes. Si la solution optimale du problème relaxé est une solution valide pour le problème original, alors elle correspond à l'optimum recherché. Sinon, des contraintes du problème initial sont violées par la solution courante. Il s'agit alors de trouver un sous-ensemble de ces contraintes appelées coupes, puis de les ajouter à la relaxation avant de la résoudre à nouveau. La procédure se poursuit jusqu'à ce que la solution obtenue soit réalisable pour le problème original, ou que les procédures d'identification des coupes n'arrivent plus à trouver de contraintes violées. Dans ce dernier cas, le coût de la dernière solution trouvée est une borne inférieure du coût optimal pour un problème de minimisation (respectivement une borne supérieure pour un problème de maximisation).

L'algorithme *Branch and cut* utilise des méthodes de coupes pour calculer de bonnes bornes inférieures au cours de la procédure d'évaluation de la méthode de séparation et évaluation afin d'améliorer ses performances.

5.2.1.4 La méthode de génération de colonnes

La méthode de *génération de colonnes* [110, 13] est une méthode efficace pour résoudre des programmes linéaires pour lesquels le nombre de variables (colonnes) et trop important pour qu'on puisse les représenter de manière explicite. L'objectif est de résoudre un problème réduit avec un ensemble limité de variables. Le problème initial est appelé problème maître, et le problème réduit est appelé problème restreint. Le problème restreint est plus simple à résoudre, mais si l'ensemble de ses variables ne contient pas celles qui donnent la solution optimale pour le programme maître, il faut rajouter au problème restreint des variables pouvant être utilisées pour améliorer la solution. Un sous problème associé au problème maître consiste

à chercher la meilleure variable à rajouter dans le problème restreint : il doit trouver la variable la plus prometteuse pour améliorer la solution.

La méthode de génération de colonnes peut être combinée avec un processus de séparation et évaluation pour résoudre un PLNE, donnant alors naissance à la méthode appelée *Branch and price* [145].

5.2.1.5 La programmation dynamique

La *programmation dynamique* repose sur le principe d'optimalité de BELLMAN [23] selon lequel une séquence optimale est composée de sous-séquences optimales. La solution optimale d'un problème peut alors être déduite en combinant des solutions optimales d'une série de sous-problèmes. On commence par les sous-problèmes les plus petits et on remonte vers les sous-problèmes de plus en plus difficiles en tirant profit des résultats des problèmes déjà obtenus. La difficulté réside dans la formulation d'une séquence de décisions permettant de calculer la solution du problème en fonction des solutions des sous-problèmes. Son efficacité dépend de la propriété récursive et de l'efficacité des procédures de résolution des problèmes intermédiaires.

5.2.1.6 La programmation par contraintes

Dans l'absolu, la *programmation par contraintes* (PPC) se situe au même niveau qu'un CSP et englobe à la fois des méthodes de résolution exactes et approchées (e.g. Constraint-Based Local Search [272], CBLS). Par abus de langage, nous réduisons la PPC au champ des méthodes de résolution complètes.

La programmation par contraintes (Constraint Programming) [16, 97, 9, 233] est une approche très populaire qui repose sur le formalisme CSP avec une recherche arborescente incluant des notions liées à la consistance locale des contraintes : elle effectue une recherche énumérative complète en inférant des réductions de l'espace de recherche à partir des contraintes du problème. La *recherche* réalise un parcours de l'espace de recherche par un jeu d'affectations et de retours arrières [169]. Souvent représentée par un arbre, la recherche énumère toutes les instanciations possibles jusqu'à trouver une solution ou conclure qu'il n'y en a pas. La *propagation* quant à elle vient assister la recherche en essayant de déduire de nouvelles informations à partir de l'état courant des domaines. À partir d'une instanciation partielle, le mécanisme de propagation tente de supprimer des valeurs impossibles des domaines des variables, réduisant ainsi la taille de l'espace de recherche.

5.2.2 Panorama des méthodes incomplètes

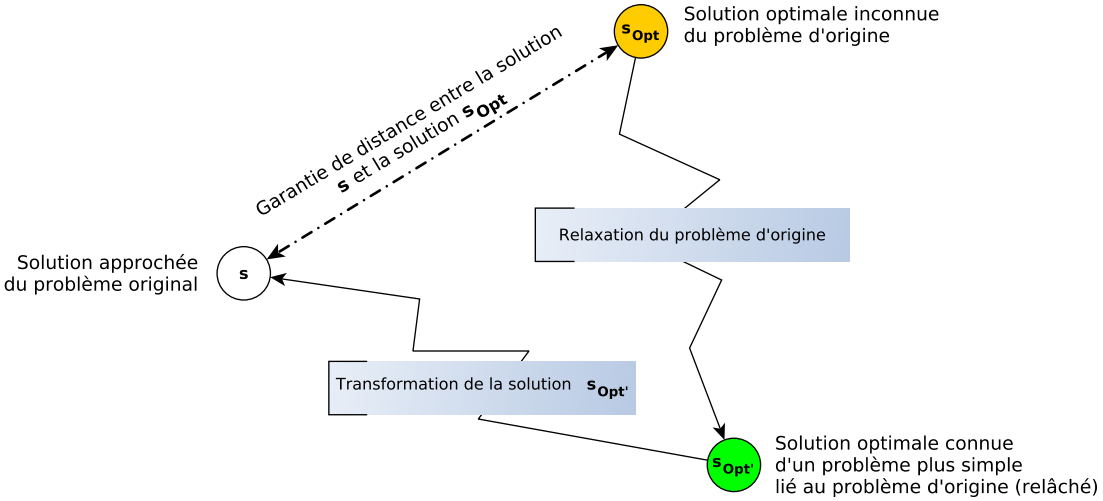
Les méthodes incomplètes reposent en grande partie sur l'utilisation d'heuristiques [2, 127]. Elles explorent une sous-partie de l'espace de recherche au moyen de techniques variées afin d'en extraire au plus vite une solution qu'on espère de bonne qualité. Contrairement aux méthodes complètes, celles-ci ne sont pas en mesure de prouver l'optimalité des solutions trouvées. L'efficacité d'une méthode incomplète réside dans l'équilibre entre deux grandes notions : l'intensification d'une part et la diversification d'autre part. L'*intensification* consiste à scruter une zone précise de l'espace de recherche afin d'en extraire un optimum local (et si possible l'optimum global) alors que la *diversification* permet de déplacer la recherche dans des zones variées et prometteuses de l'espace de recherche non encore explorées. Dans la classe des algorithmes approchés, on peut distinguer trois sous-classes :

- les algorithmes d'approximation ;
- les heuristiques ;
- les métaheuristiques ;

5.2.2.1 Les algorithmes d'approximation

L'approximation d'un problème d'optimisation $\mathcal{NP}$*-difficile* consiste à obtenir des informations sur sa solution optimale s_{Opt} sans la connaître. En général, il s'agit de résoudre un problème plus simple que le

FIGURE 5.2 – Principe lié aux algorithmes d'approximation. D'après [239].



problème original, pour transformer ensuite la solution optimale $s_{Opt'}$ du problème simplifié en une solution s du problème original (voir la figure 5.2). La difficulté est de garantir une distance maximale entre l'approximation s et l'optimum s_{Opt} du problème original. Le problème simplifié est souvent obtenu en relâchant des contraintes du problème original. Plus formellement, un *algorithme d'approximation* que l'on note ρ-approximation est un algorithme polynomial qui renvoie une solution approchée garantie au pire cas à un facteur ρ de la solution optimale [274]. Parfois, le facteur d'approximation dépend de la taille de l'instance du problème. Toutefois, pour certains problèmes $\mathcal{NP}$*-difficile*, il est impossible de faire des approximations. De plus, ces algorithmes sont spécifiques au problème ciblé et les approximations fournies sont souvent trop éloignées de la solution optimale, ce qui freine leur utilisation pour beaucoup d'applications réelles [266].

5.2.2.2 Les heuristiques

Le terme *heuristique* peut avoir différentes significations et être employé dans différents contextes, y compris par exemple dans le cadre des méthodes exactes pour le choix d'une variable ou d'une valeur particulière à évaluer (e.g. dans un arbre de décision). Nous l'employons ici pour représenter trois notions distinctes :

- soit une classe de méthodes approchées d'optimisation conformément à la classification proposée dans la figure 5.1. Dans ce cas, les métaheuristiques représentent une déclinaison de cette famille ;
- soit des méthodes d'optimisation gloutonnes capables de construire une solution en partant d'une configuration initialement vide et en choisissant à chaque étape la meilleure composante possible à insérer dans la solution (par exemple une variable de décision et une valeur de son domaine) sans jamais remettre en cause cette décision par la suite.
- soit des algorithmes problèmes-dépendants (spécifiques) connectés aux métaheuristiques pour fournir une implémentation concrète aux méthodes d'optimisation abstraites, la partie abstraite étant représentée par les métaheuristiques.

5.2.2.3 Les métaheuristiques

Dans certaines situations, on doit chercher en un temps raisonnable des solutions de bonne qualité sans pour autant garantir l'optimalité. Les *métaheuristiques* contournent le problème d'explosion combinatoire en n'exploitant délibérément qu'une partie des combinaisons prometteuses. Par conséquent, elles peuvent

ne pas trouver la solution optimale, et encore moins prouver l'optimalité de la solution trouvée. En général, elles n'offrent pas non plus de garantie d'approximation. En contrepartie, elles permettent d'obtenir rapidement de bonnes solutions pour les problèmes de grandes tailles ou très difficiles.

Certaines métaheuristiques partent d'une solution valide qu'elles modifient légèrement à chaque itération dans le but de l'améliorer. Au lieu de compter sur une solution unique pour découvrir de meilleures solutions, certaines métaheuristiques utilisent plusieurs solutions de départ. Elles combinent alors les propriétés de différentes solutions en espérant trouver des solutions de meilleure qualité.

Alors qu'une *heuristique* est souvent définie comme une procédure spécifique exploitant au mieux la structure d'un problème dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [206], les *métaheuristiques* sont des algorithmes génériques qui peuvent être utilisés pour résoudre une grande variété de problèmes d'optimisation et d'instances différentes. Elles peuvent être vues comme des méthodes générales de niveau supérieur (méta) utilisées pour guider la stratégie de conception d'heuristiques sous-jacentes, ces dernières étant adaptées au problème d'optimisation à résoudre. On retrouve au niveau des métaheuristiques les méthodes basées sur une solution unique qui manipulent et transforment une seule solution pendant le processus de recherche tandis que les algorithmes à base de population travaillent avec une population entière, c'est à dire un ensemble de solutions qui évolue au fur et à mesure des itérations [286, 266].

Métaheuristiques basées sur une solution unique Les *métaheuristiques basées sur une solution unique* regroupent les méthodes dites de recherche locale (ou à base de voisinages). Les méthodes de recherche locale (RL) sont des algorithmes itératifs qui, à partir d'une solution de départ complète, explorent l'espace de recherche en se déplaçant pas à pas d'une solution à une autre. Dans cette catégorie, nous pouvons citer parmi les plus populaires les méthodes de descente, la recherche locale itérée (Iterated Local Search, ILS) [177], le recuit simulé (Simulated Annealing, SA) [153, 42] et la recherche tabou (Tabu Seach, TS) [111, 112, 113, 115, 125].

Métaheuristiques basées sur une population de solutions Les *métaheuristiques basées sur une population de solutions* utilisent la population comme facteur de diversité et font intervenir plusieurs stratégies d'évolution de cette population conduisant à des méthodes telles que les algorithmes génétiques (Genetic Algorithm) [133, 118], la recherche par dispersion (Scatter Search, SS) [114, 116], l'algorithme des colonies de fourmis (Ant Colony Optimization, ACO) [78] ou encore l'optimisation par essaims particulaires (Particle swarm optimization, PSO) [150].

5.3 Discussion sur le choix d'une méthode de résolution

Nous savons qu'aucune méthode ne surclasse toutes les autres pour tous les problèmes. La complexité donne une première indication sur la difficulté d'un problème. Lorsque le problème à résoudre est $\mathcal{NP}$*-difficile*, une méthode complète peut nécessiter un temps de calcul qui croît de façon exponentielle avec la taille du problème à traiter [127]. Dans ce cas, l'utilisation d'une méthode incomplète peut s'avérer nécessaire pour obtenir une solution approchée de bonne qualité en un temps raisonnable. Néanmoins, bien d'autres facteurs doivent être pris en compte comme la taille des instances, leur structure, le nombre de solutions admissibles, la rapidité de calcul de la fonction d'évaluation, etc.

Bien évidemment, compte tenu du nombre très important de méthodes de résolution existantes, il n'est pas envisageable d'implémenter chaque méthode pour sélectionner la meilleure !

Pour Marc SEVAUX [244], lorsqu'on est face à un problème d'optimisation difficile, on commence par réaliser la modélisation du problème qu'on essaie ensuite de transformer en un problème de programmation mathématique pour tenter de le résoudre. La programmation mathématique permet alors d'obtenir des bornes sur le problème et permet également de mieux appréhender sa difficulté. Une fois les bornes obtenues avec quelques solutions éventuelles, on peut commencer à mettre en œuvre des métaheuristiques,

notamment des techniques de recherche locale[6] (appartiennent à la catégorie : Single-solution based Metaheuristics).

Nous verrons dans la suite de ce document que nous avons implémenté plusieurs méthodes caractéristiques de ces différentes familles sur certaines instances de notre problème. Pour obtenir des bornes et d'éventuelles solutions, nous avons commencé par évaluer une méthode complète sur notre problème. Ayant alors une meilleure vision de sa difficulté, nous avons enchaîné sur des méthodes de résolution incomplètes basées à la fois sur des heuristiques de construction (gloutonnes) pour générer des solutions initiales et sur des métaheuristiques pour améliorer le plus rapidement possible ces solutions initiales.

Néanmoins, compte tenu de la complexité de notre modèle, de sa mise au point incrémentale au fur et à mesure des réunions d'analyse avec les urbanistes et des échéances imposées pour finaliser le projet de recherche SUSTAINS, il était essentiel que ces méthodes puissent reposer sur un formalisme unique, expressif, facile à modifier et capable d'intégrer de nouvelles contraintes en fonction des nouvelles demandes des utilisateurs.

Le panorama des méthodes de résolution que nous venons d'aborder montre que les techniques d'exploration arborescente ou de programmation linéaire permettent de calculer des solutions optimales. Mais pour être efficaces sur de grosses instances, ces méthodes de résolution doivent être couplées à des techniques sophistiquées (méthodes de coupes, branch and cut, génération de colonnes, branch and price, ...). On constate également que le formalisme relatif à ces méthodes s'adapte difficilement à de nouvelles contraintes [267]. À travers cette étude très générale, il apparaît également que le formalisme CSP réputé pour sa grande souplesse et son expressivité peut être exploité à la fois par les méthodes complètes de programmation par contraintes et par les métaheuristiques de recherche locale, cette dernière approche ayant donné naissance à la *Constraint-Based Local Search* [272].

Compte tenu de cette proximité entre PPC et RL à travers le formalisme CSP, nous proposons de mettre en perspective, dans les sections suivantes, ces différents procédés en insistant plus particulièrement sur les métaheuristiques de recherche locale qui regroupent de nombreuses méthodes. L'inventaire des principales techniques nous permettra alors d'identifier les méthodes de RL les plus appropriées à nos problématiques.

Mais tout d'abord, soulignons que pour résoudre des problèmes industriels de plus en plus complexes, la tendance actuelle est à l'hybridation de différentes méthodes pour s'appuyer sur les points forts de différentes techniques sur le principe qui pourrait s'apparenter à : "*l'union fait la force*". Une autre voie consiste à utiliser la parallélisation, avec une décomposition qui intervient soit au niveau des traitements (algorithmes), soit au niveau de l'espace de recherche, soit sur l'évaluation de la fonction objectif, cette fois sur le principe : "*diviser pour mieux régner*". Notons que ces deux approches ne sont pas exclusives et qu'elles peuvent tout à fait cohabiter.

5.4 Programmation par contraintes

En programmation par contraintes (PPC), un problème est défini par un CSP (voir définition 4 page 58) permettant de représenter un grand nombre de problèmes. Par exemple, une contrainte "$x + 3{*}y = 12$" qui restreint les valeurs que l'on peut affecter simultanément aux variables x et y.

Il existe de nombreuses méthodes pour résoudre les CSP :

- les méthodes de résolution complètes dont les plus élaborées exploitent les notions de consistance, de filtrage et de propagation ;
- les méthodes de résolution incomplètes représentées par les métaheuristiques (i.e. Constraint-Based Local Search [272, 47]) ;
- les méthodes hybrides qui combinent des méthodes complètes et incomplètes [76].

Nous décrivons ici les méthodes de résolution complètes basées sur les notions de consistance et faisant appel aux techniques de filtrage et de propagation de contraintes.

[6] On parlera indifféremment de "*méthodes de recherche locale*" ou de "*métaheuristiques à base de voisinages*".

Résoudre un CSP consiste à trouver une affectation pour chaque variable satisfaisant l'ensemble des contraintes. La notion de consistance repose sur la satisfaction des contraintes afin de réduire l'espace de recherche pour rendre moins difficile la résolution du problème [76].

5.4.1 Principes de la PPC

La PPC utilise une approche de construction qui bâtit pas à pas une solution en partant d'une solution partielle initialement vide qu'elle cherche à étendre à chaque étape. Pour cela, elle détermine la prochaine variable, choisit une valeur dans son domaine et l'ajoute pour obtenir un nouvelle solution partielle, ce processus étant répété jusqu'à ce que l'on obtienne une solution complète (ou la preuve qu'il n'y a pas de solution). Durant la recherche d'une solution, une méthode de construction fait intervenir des heuristiques pour effectuer (i) le choix de la variable et (ii) le choix de la valeur à affecter à cette variable. La programmation par contraintes s'articule autour de quatre éléments majeurs :

- le CSP,
- les algorithmes de filtrage,
- la propagation,
- la recherche de solutions.

5.4.1.1 CSP (ou réseau de contraintes)

Étant donné le CSP $P = (X, D, C)$, la résolution de P consiste à affecter des valeurs aux variables, de telle sorte que toutes les contraintes soient satisfaites. Une solution est une "affectation totale consistante", c'est-à-dire une valuation de toutes les variables du problème qui ne viole aucune contrainte.

Lorsqu'un CSP n'a pas de solution, on dit qu'il est *surcontraint* : il y a trop de contraintes et on ne peut pas toutes les satisfaire. Dans ce cas, on peut souhaiter trouver l'affectation totale qui viole le moins de contraintes possibles. Un tel CSP est appelé *max-CSP*, car on cherche alors à maximiser le nombre de contraintes satisfaites. D'autres formalismes ont été proposés parmi lesquelles les CSP valués (VCSP) [241]. Le formalisme VCSP introduit une graduation dans la violation des contraintes. Une valeur (appelée *valuation*) est associée à chaque contrainte, cette valuation traduisant l'importance de la violation de la contrainte. Une solution du problème est une affectation totale qui peut éventuellement violer certaines contraintes et dont l'importance des violations est minimale suivant un critère et un ordre donnés.

Inversement, lorsqu'un CSP admet beaucoup de solutions différentes, on dit qu'il est *sous-contraint*. Si les différentes solutions ne sont pas toutes équivalentes, dans le sens où certaines sont mieux que d'autres, on peut exprimer des préférences entre les différentes solutions via un CSOP (voir définition 8). L'objectif consiste alors à trouver une solution qui minimise (respectivement qui maximise) la fonction objectif liée au CSOP.

5.4.1.2 Filtrage

La programmation par contraintes utilise pour chaque sous-problème une méthode de résolution spécifique à ce sous-problème afin de supprimer les valeurs des domaines des variables impliquées dans le sous-problème qui, compte tenu des valeurs des autres domaines, ne peuvent appartenir à aucune solution de ce sous-problème. Ce mécanisme est appelé *filtrage* (voir algorithme 1). En procédant ainsi pour chaque sous-problème, donc pour chaque contrainte, les domaines des variables vont se réduire, réduisant par la même occasion l'espace de recherche associé au problème. Un algorithme de filtrage (ou de réduction de domaines) est associé à chaque contrainte. Par exemple, pour la contrainte $(x < y)$ avec $D(x) = [10; 20]$ et $D(y) = [0; 15]$, un algorithme de filtrage associé à cette contrainte pourra supprimer les valeurs de 15 à 20 de $D(x)$ et les valeurs de 0 à 10 de $D(y)$. Un algorithme de filtrage associé à une contrainte binaire (qui porte sur deux variables) réalise la consistance d'arc si il supprime toutes les valeurs des variables impliquées dans la contrainte qui ne sont pas consistantes avec la contrainte. Par exemple, pour la contrainte $x + 3 = y$ avec

les domaines $D(x) = \{1, 3, 4, 5\}$ et $D(y) = \{4, 5, 8\}$, un algorithme de filtrage établissant la consistance d'arc modifiera les domaines pour obtenir $D(x) = \{1, 5\}$ et $D(y) = \{4, 8\}$.

Algorithme 1 : Algorithme naïf de filtrage d'après la version proposée par Philippe DAVID (http://www.emn.fr/z-info/pdavid/Enseignement/IA/poly-ia/csp/arc-consistance.html).

```
Fonction Revise (x, y)
begin
1   SUPPRESSION ← Faux ;
2   forall (valeur val_x de D(x)) do
3       if (il n'existe pas de valeur val_y dans D(y) support de val_x) then
4           Supprimer val_x de D(x) ;
5           SUPPRESSION ← Vrai ;
6   return SUPPRESSION ;
end
```

5.4.1.3 Propagation

Après chaque modification du domaine d'une variable, il est utile de réétudier l'ensemble des contraintes impliquant cette variable car cette modification peut conduire à de nouvelles déductions. Autrement dit, la réduction du domaine d'une variable peut permettre de déduire que certaines valeurs d'autres variables n'appartiennent pas à une solution. Ce mécanisme est appelé *propagation*. Dès lors qu'un algorithme de filtrage associé à une contrainte modifie le domaine d'une variable, les conséquences de cette modification sont étudiées pour les autres contraintes impliquant cette variable : les algorithmes de filtrage des autres contraintes sont appelés afin de déduire éventuellement d'autres suppressions. On dit alors qu'une modification a été propagée. Ce mécanisme de propagation est répété jusqu'à ce que plus aucune modification n'apparaisse.

5.4.1.4 Recherche de solutions

Afin de parvenir à une solution, l'espace de recherche va être parcouru en essayant d'affecter successivement une valeur à toutes les variables. Les mécanismes de filtrage et de propagation étant bien entendu relancés après chaque essai puisqu'il y a modification de domaines. Parfois, une affectation peut entraîner la disparition de toutes les valeurs d'un domaine : on dit alors qu'un échec se produit ; le dernier choix d'affectation est alors remis en cause, il y a "backtrack" ou "retour en arrière" et une nouvelle affectation est tentée (voir algorithme 2). Une méthode de *retour arrière* avec une stratégie de recherche en profondeur d'abord consiste à fixer à chaque étape la valeur d'une variable. Aussitôt qu'un échec est détecté, un retour arrière est effectué, i.e., une ou plusieurs instanciations déjà effectuées sont annulées et de nouvelles valeurs recherchées [29]. Par exemple, un algorithme typique avec retour arrière pour la résolution d'un problème de satisfaction de contraintes cherche à prolonger à chaque étape l'assignation courante de manière consistante. En cas d'échec, un retour arrière est effectué sur la dernière variable instanciée possédant encore des valeurs non essayées [179]. Les méthodes avec retour arrière sont en général complètes et de complexité exponentielle. Pour réduire le nombre de retour arrière (et le temps de recherche), on utilise les techniques de filtrage décrites précédemment afin d'anticiper le plus tôt possible les échecs.

5.4.2 Complément sur les contraintes

Une contrainte est une relation entre différentes variables. Cette relation peut être définie en *extension* en énumérant les tuples de valeurs appartenant à la relation ($x = 0$ et $y = 1$) ou en *intension* en utilisant des

Algorithme 2 : Backtrack (version rudimentaire) d'après la version proposée par Philippe DAVID (http://www.emn.fr/z-info/pdavid/Enseignement/IA/poly-ia/csp/backtrack.html).

Procédure Backtrack(V-ins, var-courante, V-non-ins, val)

begin

1 var-courante ← val ;
2 **if** *(l'instanciation courante est consistante)* **then**
3 **if** *(V-non-ins est vide)* **then**
4 l'instanciation courante est une solution ;
else
5 soit y la prochaine variable à instancier
6 **forall** *(valeur* val_y *de* $D(y)$*)* **do**
7 faire Backtrack(V-ins $\cup \{$var-courante$\}$, y, V-non-ins $-\{y\}$, val_y) ;

end

propriétés mathématiques connues ($x < y$). En fonction du nombre de variables sur lesquelles elle porte (arité), on dira qu'une contrainte est *unaire* si elle porte sur une seule variable ($x*x = 4$), *binaire* si elle met en relation deux variables ($x \neq y$) ou encore *n-aire* si elle met en relation un ensemble de n variables. On distingue différents types de contraintes en fonction des domaines de valeurs des variables. Il y a notamment les contraintes numériques (sur les réels, les entiers) et les contraintes booléennes. Les contraintes peuvent être exprimées sous différentes formes : formules mathématiques, tables de valeurs compatibles, etc.

5.4.3 Résolution des DCSP

Ayant résolu un problème P, il s'agit de résoudre un problème P' dérivé de P par l'ajout d'un ensemble de contraintes C_a et le retrait d'un ensemble de contraintes C_r. Ce qui revient finalement à résoudre une succession de problèmes statiques et permet au passage de s'appuyer sur les nombreux travaux existants relatifs à la résolution de ces derniers.

5.4.4 Résolution des CSOP

Dans le cas de la résolution d'un CSOP, on veut à la fois (i) trouver une solution qui satisfait l'ensemble des contraintes donné et (ii) optimiser une fonction définissant un objectif. Le but est de trouver une solution pour $P = (X, D, C)$ qui minimise (respectivement qui maximise) la fonction objectif f. Pour la résolution des problèmes d'optimisation, on distinguera donc deux types de solutions :

- les solutions du problème de satisfaisabilité, c'est-à-dire celles qui ne tiennent pas compte de la fonction f,
- les solutions optimales, c'est-à-dire les solutions du problème de satisfaisabilité qui minimisent (respectivement qui maximisent) la fonction objectif f.

Definition 9 (Optimum global)**.** Pour un problème de minimisation, une solution $s^* \in \mathcal{S}$ est un optimum global (ou solution optimale) si $f(s^*) \leq f(s)$ pour tout élément $s \in \mathcal{S}$. (et respectivement $f(s^*) \geq f(s)$ pour un problème de maximisation).

Résoudre à l'optimalité un problème consiste à trouver une solution s^* correspondant à un optimum global. Pour la résolution d'un CSOP, nous supposons qu'une variable x est ajoutée à la liste des variables X de P avec $x = f(X)$. L'approche la plus intuitive est de trouver une solution optimale par la résolution

d'une séquence de CSP. Initialement, un algorithme de backtracking[7] est utilisé pour trouver une solution qui satisfait toutes les contraintes. Chaque fois qu'une solution s est trouvée, une contrainte s'ajoute au CSP avec la forme $x > f(s)$ qui exclut toutes les solutions qui ne sont pas meilleures que s pour un problème de minimisation (respectivement $x < f(s)$ pour un problème de maximisation). Ce processus est répété jusqu'à ce que le CSP résultant soit infaisable ; dans ce cas la dernière solution trouvée est optimale.

5.4.5 Conclusion

La PPC permet de résoudre une grande variété de problèmes allant des problèmes de satisfaction de contraintes aux problèmes d'optimisation combinatoire en passant par les problèmes dynamiques. Pour cela, elle a su s'adapter à des variantes du CSP (VCSP, DCSP, CSOP, ...) imaginées par une communauté scientifique dynamique afin de représenter ces différentes catégories de problèmes. Pour s'éviter une énumération exhaustive de toutes les configurations possibles qui lui serait fatale, elle met en œuvre des procédés élaborés basés sur la consistance pour réaliser un parcours intelligent de l'espace de recherche. Toutes ces qualités font de la PPC un paradigme de programmation de premier plan pour résoudre les nombreux problèmes liés à nos sociétés modernes.

5.5 Heuristiques de construction (approchées)

Les méthodes de construction produisent de façon incrémentale des solutions admissibles en partant d'une configuration initialement vide. Elles insèrent à chaque étape une composante dans la configuration partielle courante jusqu'à obtenir une solution complète. Elles opèrent ainsi, pas à pas, sans tenir compte de toutes les conséquences sur le coût de la solution finale. A ce titre, elles sont souvent considérées comme des méthodes "myopes".

Elles reposent sur une approche "basée sur les modèles" car elles utilisent un modèle pour choisir à chaque itération la prochaine composante à ajouter à la configuration partielle [294]. A chaque étape, le modèle doit déterminer le choix de la variable suivante et le choix de la valeur pour cette variable. Les méthodes de cette classe diffèrent entre elles selon les modèles utilisés et la performance de ces méthodes dépend largement de leur capacité à exploiter la structure du problème.

Les méthodes de construction se distinguent par leur rapidité et leur grande simplicité. Néanmoins, leur manque de visibilité sur la solution finale se traduit en général par des solutions produites de faible qualité.

Notre étude porte plus spécifiquement sur les méthodes de construction "gloutonnes" qui sont les plus connues et les plus simples à mettre en œuvre. A noter toutefois que les algorithmes par estimation de distributions (*Estimation of distribution algorithms*, EDA) [164] et l'optimisation par colonies de fourmis viennent compléter cette catégorie d'heuristiques incomplète.

5.5.1 Algorithmes gloutons

En 1971, J. EDMONDS a introduit pour la première fois dans la littérature une étude sur les heuristiques gloutonnes dans le cadre de l'optimisation combinatoire [85].

Les algorithmes gloutons (*Greedy algorithm*) choisissent à chaque étape un composant à insérer dans la configuration partielle[8] pour lequel une heuristique donnée produit un coût minimal (problème de minimisation). Contrairement aux méthodes exactes avec retour arrière (backtracking), les choix opérés ne sont jamais remis en cause par la suite. La qualité de la solution produite est directement liée à l'heuristique de choix utilisée et les premières décisions prises influencent largement la qualité de la solution finale lorsque

[7] Algorithme à essais successifs qui explore l'arbre des solutions du problème combinatoire en élagant les branches n'aboutissant pas à une solution et remettant en cause les choix antérieurs lorsqu'on atteint un cul-de-sac (une feuille de l'arbre non solution du problème).

[8] On fixe la valeur d'une variable de décision libre (c'est à dire sans affectation).

les variables du problème sont liées. Pour certains problèmes (recherche d'un arbre recouvrant de poids minimum avec l'algorithme de KRUSKAL), un algorithme glouton conduit à la solution optimale, mais c'est rarement le cas.

Algorithme 3 : Algorithme glouton déterministe [267]

```
1  echec ← faux ;
2  c ← ∅ (configuration partielle initialement vide) ;
3  while (c incomplète et echec = faux) do
4      Construire la liste L des éléments insérables dans c ;
5      if (L≠∅) then
6          Évaluer le coût incrémental des éléments de L ;
8          Insérer dans c l'élément de L ayant le coût incrémental le plus faible ;
       else
9          echec ← vrai ;
10 return c ou echec ;
```

Selon l'algorithme 3, la configuration c initialement vide peut être vue comme un ensemble d'éléments (i.e. de variables du problème dont les valeurs sont fixées). A chaque itération, on définit un ensemble fini $\mathcal{L} = \{e_1, e_2, \cdots, e_n\}$ constitué d'éléments pouvant être insérés dans la configuration partielle c. Une fonction objectif définie comme $f : \mathcal{L} \rightarrow \mathbb{R}$ permet d'évaluer le coût incrémental de chaque élément de $\mathcal{L}$. L'élément de $\mathcal{L}$ ayant le coût incrémental le plus faible (i.e. celui qui engendre la plus petite augmentation du coût de c) est inséré dans c. Lorsqu'un élément e_i est sélectionné pour faire partie de la solution, il n'est jamais remplacé par un autre élément. Les algorithmes gloutons sont des algorithmes déterministes [266].

5.5.2 Algorithmes gloutons aléatoires

Les algorithmes gloutons aléatoires (*Greedy randomized algorithm*) peuvent construire plusieurs configurations et reposent sur une stratégie gloutonne pour le choix des composants à intégrer à chaque itération dans les configurations en cours de construction. Néanmoins, l'étape de fixation de la variable courante et de sa valeur est légèrement modifiée afin de permettre plusieurs choix plutôt qu'un seul à chaque itération. Les choix potentiels constituent une liste (Restricted Candidate List, RCL) dans laquelle un candidat sera retenu aléatoirement. Une fois le couple (variable, valeur) fixé, cette liste est mise à jour en tenant compte de la configuration partielle courante. Cette étape est itérée jusqu'à l'obtention d'une configuration complète [245].

Une première façon de procéder consiste à choisir aléatoirement le prochain composant parmi les k meilleurs. Il est aussi possible de faire un choix aléatoire parmi les composants éloignés d'au plus un certain pourcentage par rapport au meilleur composant trouvé. Enfin, une autre option consiste à choisir le prochain composant en fonction de probabilités liées à la qualité des différents composants [143].

5.5.3 Conclusion

Les algorithmes gloutons permettent de construire très rapidement des solutions, ce qui constitue déjà un résultat intéressant pour des problèmes contraints ou/et de très grande taille. Ces procédures sont relativement simples à concevoir et reposent sur la structure du problème à considérer. Mais les résultats obtenus grâce à ces techniques sont souvent de faible qualité.

Toutes ces caractéristiques font que les algorithmes gloutons sont des partenaires idéals pour les métaheuristiques à base de voisinage qui ont besoin d'une solution initiale pour pouvoir fonctionner.

5.6 Métaheuristiques à base de voisinages

Le terme métaheuristique a été introduit pour la première fois par F. GLOVER en 1986 [111]. Les métaheuristiques permettent de s'attaquer à des instances difficiles de problèmes d'optimisation variés tout en délivrant des solutions satisfaisantes en un temps raisonnable. En contrepartie, elles n'offrent pas la garantie de trouver les solutions optimales ni même des solutions dont la qualité serait bornée. Les métaheuristiques bénéficient d'un intérêt croissant depuis plus d'une vingtaine d'années et elles s'appliquent avec succès à une large gamme de problèmes complexes et de grande taille. Beaucoup de critères différents peuvent être utilisés pour regrouper les métaheuristiques :

- inspirée ou non de la nature ;
- avec ou sans mémoire ;
- déterministe ou stochastique ;
- basée sur un solution unique ou sur une population ;
- itérative ou gloutonne.
- etc.

Dans ce manuscrit, nous nous appuyons sur la classification qui distingue celles qui exploitent une "*population de solutions*" et celles qui utilisent une "*solution unique*" pendant le processus de résolution. Ce chapitre concerne plus précisément les métaheuristiques qui exploitent une solution unique, que l'on retrouve également sous le nom de "*métaheuristiques à base de voisinages*" ou "*recherche locale*".

5.6.1 Généralités sur les métaheuristiques

5.6.1.1 Propriétés

Une métaheuristique est constituée d'un ensemble de concepts fondamentaux qui permet d'aider à la conception de méthodes heuristiques pour un problème d'optimisation. Ainsi, une métaheuristique est adaptable et applicable à une large classe de problèmes. C. BLUM et A. ROLI ont proposé dans [30] un ensemble de propriétés intéressantes qui caractérisent les métaheuristiques :

- les métaheuristiques sont des stratégies qui permettent de guider la recherche d'une solution optimale.
- le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions (presque) optimales.
- les techniques qui constituent des algorithmes de type métaheuristique vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.
- les métaheuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité.
- les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.
- les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, sans faire référence à un problème spécifique.
- les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.
- les métaheuristiques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum pour mieux guider la suite du processus de recherche.

Ces propriétés définissent le comportement de toutes les métaheuristiques pendant la recherche d'une solution, en allant de la plus simple à la plus complexe.

5.6.1.2 Quand utiliser une métaheuristique ?

Il n'est pas judicieux d'utiliser une métaheuristique pour résoudre des problèmes lorsqu'il existe des algorithmes exacts qui sont efficaces pour le faire. Néanmoins, différentes situations justifient l'utilisation d'une métaheuristique :

- un problème facile (appartenant à la classe P) avec de très grosses instances. Dans ce cas, des algorithmes exacts en temps polynomial sont connus mais ils peuvent être trop coûteux compte tenu de la taille des instances.
- un problème facile avec une contrainte forte de temps réel. Pour des problèmes d'optimisation temps réel[9], les métaheuristiques sont largement utilisées. Dans cette classe de problèmes, nous devons trouver une bonne solution en ligne. Même si il existe des algorithmes exacts efficaces pour résoudre le problème, les métaheuristiques sont utilisées pour réduire le temps de recherche. Les problèmes d'optimisation dynamique représentent un exemple de tels problèmes.
- un problème difficile (appartenant à la classe $\mathcal{NP}$-*difficile*) avec des instances de taille modérée et/ou des structures difficiles.
- des problèmes d'optimisation avec une fonction objectif et/ou des contraintes qui nécessitent beaucoup de temps de calcul. Certains problèmes d'optimisation réels sont caractérisés par un temps de calcul énorme de la fonction objectif.
- des modèles non analytiques de problèmes d'optimisation qui ne peuvent pas être résolus de façon exhaustive : beaucoup de problèmes pratiques sont définis par une boîte noire noyée dans une fonction objectif.

5.6.1.3 Définition du mot métaheuristique

Il existe beaucoup de tentatives pour définir ce qu'est une *métaheuristique* sans que l'on puisse toutefois trouver une définition qui fasse vraiment l'unanimité.

Certaines se concentrent sur l'aspect problème-indépendant comme celle proposée sur le site "the metaheuristics network"[10] : "*Une métaheuristique représente un ensemble de concepts utilisés pour définir des méthodes heuristiques pouvant être appliquées à un large ensemble de problèmes différents. En d'autres termes, une métaheuristique peut être vue comme un cadre algorithmique général qui peut être appliqué à différents problèmes d'optimisation avec relativement peu de modifications pour les adapter à un problème spécifique*".

Dans [280], Voß et al. se concentrent quant à eux sur la distinction entre une stratégie de haut niveau et des mécanismes de bas niveau : "*Une métaheuristique est un processus itératif maître qui guide et modifie les opérations des heuristiques subordonnées pour produire efficacement des solutions de haute qualité. Elle peut manipuler une solution unique complète (ou incomplète) ou une collection de solutions à chaque itération. Les heuristiques subordonnées peuvent être des procédures de haut ou de bas niveau, ou une simple recherche locale, ou juste une méthode de construction*".

OSMAN et LAPORTE insistent sur l'*intensification* et la *diversification* aussi bien que sur la nature itérative du processus de recherche [210] : "*Une métaheuristique est formellement définie comme un processus de génération itératif qui guide une heuristique subordonnée en combinant intelligemment différents concepts pour l'exploration et l'exploitation de l'espace de recherche, des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions quasi-optimales*".

Parmi ces dernières, nous retiendrons pour définir une métaheuristique (i) l'aspect problème-indépendant qui offre un cadre générique applicable à tout type de problème et (ii) l'équilibre nécessaire entre l'intensification et la diversification qui représente un aspect essentiel de toute métaheuristique.

5.6.1.4 Gestion des contraintes

Beaucoup de problèmes d'optimisation sont liés à des contraintes dures qu'on ne peut pas réduire dans la fonction objectif. Pour permettre aux métaheuristiques de gérer efficacement ces contraintes, différentes stratégies ont été mises au point [190, 212, 79, 266] :

[9] Les logiciels embarqués dans les récepteurs GPS (Global Positioning System) utilisent des heuristiques pour trouver un chemin entre deux points compte tenu du nombre important de nœuds et de la nécessité de fournir une réponse en temps réel.

[10] http://www.metaheuristics.org/

- rejet ;
- pénalité ;
- réparation ;
- décodage ;
- préservation.

Stratégie de rejet La stratégie de rejet est très simple : seules les solutions faisables sont conservées durant la recherche et les solutions infaisables sont automatiquement jetées. Cette stratégie est concevable si la portion des solutions infaisables de l'espace de recherche est réduite.

Stratégie de pénalité Avec la stratégie de pénalité, les solutions infaisables sont prises en compte pendant le processus de recherche. La fonction objectif non contrainte est étendue par une fonction de *pénalité* qui se chargera de pénaliser les solutions infaisables. C'est l'approche la plus populaire. Il existe beaucoup d'alternatives pour définir les pénalités [105].

Stratégie de réparation Les stratégies de réparation mettent en œuvre des algorithmes heuristiques (procédure de réparation) qui transforment une solution infaisable en solution faisable [157]. Ces stratégies spécifiques au problème sont employées lorsqu'un opérateur de recherche peut générer des solutions infaisables.

Stratégie de décodage Cette stratégie utilise des encodages indirects qui transforment la topologie de l'espace de recherche. Lorsqu'on utilise une représentation indirecte, l'encodage n'est pas une solution complète du problème. Un décodeur est nécessaire pour exprimer la solution donnée par l'encodage. Selon l'information présente dans l'encodage, le décodeur aura plus ou moins de travail à faire pour dériver une solution complète. Le décodeur peut être non déterministe. Les contraintes associées au problème d'optimisation peuvent être gérées par le décodeur qui garantira alors la validité de la solution dérivée.

La procédure de décodage peut être vue comme une fonction $S \rightarrow \mathcal{S}$ qui associe à chaque configuration $c \in S$ une solution faisable $s \in \mathcal{S}$ dans l'espace de recherche. La fonction de décodage doit disposer des propriétés suivantes [63] :

- à chaque configuration $c \in S$ correspond une solution faisable $s \in \mathcal{S}$;
- pour chaque solution faisable $s \in \mathcal{S}$, il y a une configuration $c \in S$ qui lui correspond ;
- la complexité de calcul du décodeur doit être réduite ;
- les solutions faisables dans $\mathcal{S}$ doivent avoir le même nombre de configurations associées dans S ;
- l'espace de représentation doit posséder la propriété de localité dans le sens que la distance entre les configurations de S doit être en corrélation avec la distance entre les solutions faisables dans $\mathcal{S}$.

Stratégie de préservation Une représentation spécifique et des opérateurs garantissent la génération de solutions faisables. Cette stratégie incorpore des connaissances spécifiques du problème dans la représentation et dans les opérateurs de recherche pour générer uniquement des solutions faisables. Cette stratégie est adaptée à des problèmes spécifiques et ne peut pas être généralisée à tous les problèmes d'optimisation sous contraintes.

5.6.1.5 Réglage des paramètres

La majorité des métaheuristiques nécessite un ajustement méticuleux de beaucoup de paramètres en fonction du problème à résoudre. Ces paramètres dont les valeurs sont parfois difficiles à définir *a priori* peuvent avoir une grande influence sur l'efficacité de la résolution. Néanmoins, il n'existe pas de valeurs optimales des paramètres pour une métaheuristique qui puissent être applicables à tous les problèmes.

Deux stratégies sont utilisées pour le réglage de ces paramètres : une initialisation *hors ligne* et une initialisation *en ligne*.

Réglage hors ligne Dans ce mode, les valeurs des différents paramètres sont fixés avant l'exécution de la métaheuristique.

Réglage en ligne Avec l'approche en ligne, les paramètres sont contrôlés et modifiés dynamiquement ou de façon adaptative pendant l'exécution de la métaheuristique. La version dynamique prend en charge le changement de valeur d'un paramètre sans prendre en compte la progression de la recherche alors que la version adaptative change les valeurs en fonction de la progression de la recherche.

Différentes implémentations ont été proposées pour adapter le paramétrage pendant la phase de résolution en fonction de l'instance. Avec l'approche *Tabou réactive* [18], on ajuste automatiquement (version dynamique) la longueur de la liste tabou. Dans [203], les auteurs proposent une procédure dédiée pour régler automatiquement (version adaptative) le paramètre « max » (i.e. nombre de voisins maximum à explorer à chaque mouvement) de la métaheuristique *ID Walk* également décrite dans leur document. Cette procédure de réglage automatique procède par apprentissage et elle a pu être appliquée avec succès à d'autres algorithmes pour définir la valeur d'un ou de deux paramètres (e.g. recuit simulé, recherche tabou).

5.6.1.6 Mesures de performance

Pour les méthodes exactes capables de garantir l'optimalité des résultats, le temps de recherche représente l'indicateur principal pour évaluer la performance des algorithmes. Pour les métaheuristiques n'offrant pas les mêmes garanties, d'autres indicateurs doivent être pris en compte [15] :

- la qualité des solutions ;
- l'effort de calcul ;
- la robustesse.

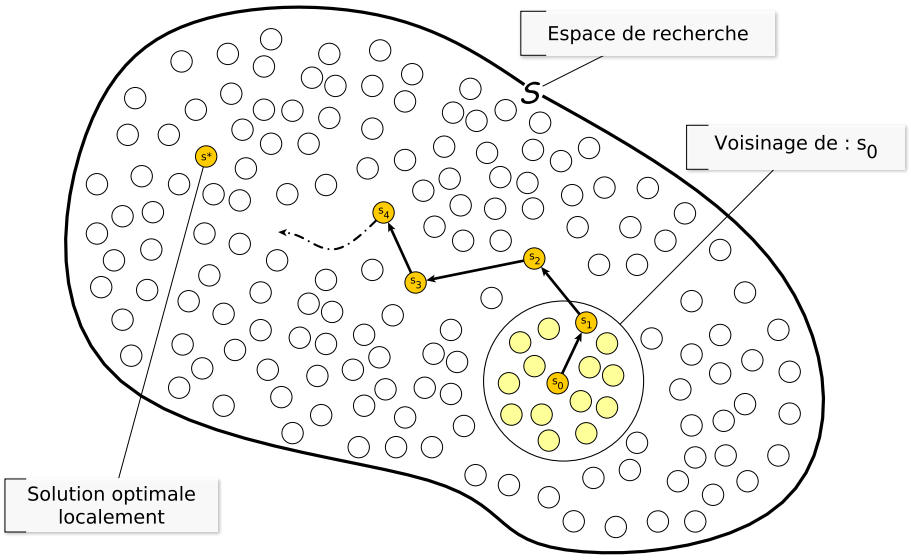
Qualité des solutions La qualité des solutions obtenues s'évalue en terme de précision en considérant une mesure de distance ou un pourcentage de déviation entre la solution obtenue et l'une des solutions suivantes : solution optimale, borne inférieure, meilleure solution connue ou seuil à atteindre.

Les solutions optimales peuvent être trouvées par des algorithmes exacts. Néanmoins, pour beaucoup de problèmes complexes, les solutions optimales ne sont pas connues. Dans ce cas, on peut utiliser des bornes inférieures pour des problèmes de minimisation (et respectivement des bornes supérieures pour des problèmes de maximisation) proches de ces valeurs optimales. Lorsqu'il s'agit de problèmes classiques, des instances standards sont généralement disponibles avec les meilleures solutions connues. Enfin, pour des instances de problèmes spécifiques, un décideur peut définir un seuil à atteindre sur la qualité de la solution pour qu'elle puisse être acceptée.

Effort de calcul Lors d'une analyse empirique, on relève le temps de calcul exprimé soit en temps CPU, soit en temps réel écoulé. Mais ces mesures ont l'inconvénient d'être liées aux caractéristiques du matériel informatique utilisé, de l'infrastructure réseau, du système d'exploitation, etc. Le nombre d'évaluations de la fonction objectif, indépendant de l'infrastructure informatique, peut également servir de mesure lorsque le temps de calcul de la fonction est significatif et reste constant. Pour comparer les effets de différents paramétrages d'une même méthode sur une instance de problème identique, on peut simplement évaluer le nombre d'itérations de la boucle principale.

Robustesse En général, la robustesse peut être définie par une insensibilité vis à vis d'écarts minimes dans les instances d'entrée ou vis à vis des paramètres utilisés : plus faible est la variation des solutions obtenues, meilleur est la robustesse [195]. Pour les métaheuristiques, la robustesse peut également être évaluée par

FIGURE 5.3 – Fonctionnement général d'une méthode de recherche locale : représentation sous la forme d'une marche à travers l'espace de recherche en partant d'une solution initiale s_0 et en se dirigeant de proche en proche vers une solution optimale localement s^*.



rapport à des problèmes ou instances différentes. Et lorsque des algorithmes stochastiques sont utilisés, la robustesse peut faire référence à des écarts de comportement de l'algorithme sur différentes exécutions pour une même instance de problème.

5.6.2 Métaheuristiques de voisinage

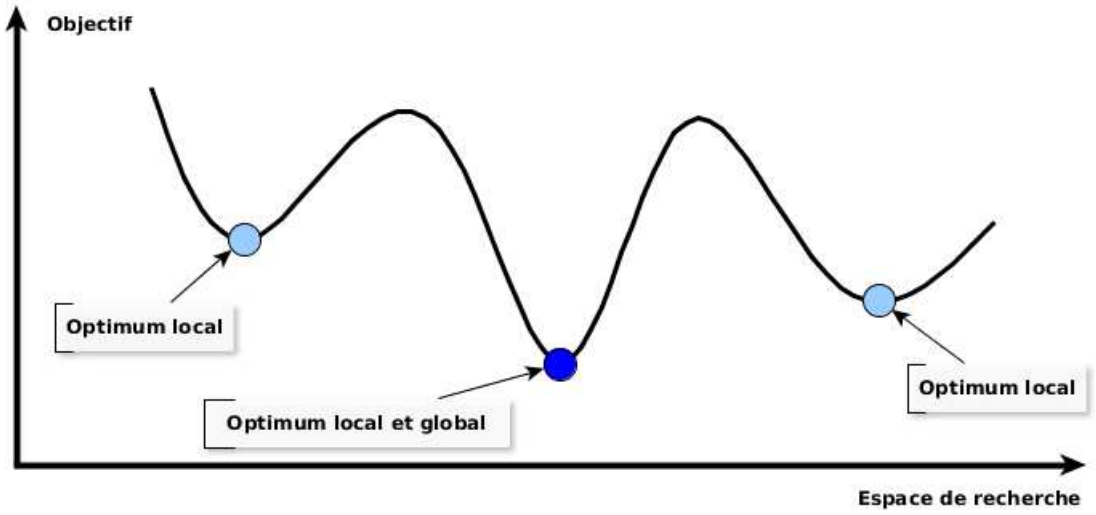
Une métaheuristique de voisinage améliore une solution unique pendant toute la phase d'optimisation. Elle peut être vue comme une *marche* de proche en proche dans les voisinages, une *trajectoire de recherche* à travers l'espace de recherche du problème [57]. La marche (ou la trajectoire) est accomplie par des procédures itératives de génération et de remplacement qui réalisent tour à tour un déplacement de la solution courante à une solution voisine dans l'espace de recherche. Dans la phase de génération, un ensemble de solutions candidates est généré à partir de la solution courante alors que dans la phase de remplacement, on effectue la sélection d'un candidat à partir de l'ensemble des solutions candidates pour remplacer la solution courante. Cette procédure se répète jusqu'à ce qu'un critère d'arrêt donné soit vérifié.

La recherche locale se base en effet sur l'idée d'amélioration d'une solution donnée s en appliquant des changements sur ses affectations. Les solutions obtenues en modifiant s s'appellent les *voisins* de s et constituent un ensemble qui représente le *voisinage* de s. Ainsi, la recherche locale commence par une solution initiale (éventuellement non-consistante) pour se diriger vers une solution optimale (ou localement optimale) :

- un *voisinage* est généré autour de la solution courante : il représente l'ensemble des solutions obtenues en altérant localement la solution courante, par exemple en changeant la valeur de l'une de ses variables ou en échangeant les valeurs de deux variables.
- une *heuristique de sélection* détermine l'un des voisins qui devient la solution courante, par exemple celui qui permet de faire diminuer le plus le coût de la solution (pour un problème de minimisation).
- un *critère d'arrêt* est testé pour limiter le temps de calcul, par exemple un nombre maximum d'itérations ou une durée de traitement maximale.

Le fonctionnement général d'une méthode de recherche locale peut être illustré par la Figure 5.3 : $\mathcal{S}$ correspond à l'espace des solutions ; $s_0, s_1, \cdots, s_4$ sont des solutions admissibles et s^* correspond à une solution optimale localement.

FIGURE 5.4 – Optimum global et local pour un problème de minimisation.



Pour un voisinage donné, si on se contente de choisir systématiquement un voisin qui réduit le coût de la solution courante (pour un problème de minimisation), le processus de résolution se retrouvera obligatoirement bloqué dans un optimum local où tous les voisins dégradent le coût de la solution courante, cet optimum local n'étant que très rarement l'optimum global (figure 5.4). C'est pour cette raison que les métaheuristiques de RL utilisent des stratégies pour s'en échapper, par exemple en autorisant des transitions défavorables (i.e. choix d'un voisin de coût supérieur) ou en recommençant la recherche avec une nouvelle solution choisie aléatoirement.

Les algorithmes de RL ne sont pas complets car ils ne maintiennent pas de représentation de la partie déjà explorée de l'espace de recherche, alors que les algorithmes systématiques le font en structurant l'espace sous la forme d'un arbre. Ils ne peuvent donc pas garantir que tout l'espace de recherche sera exploré ni qu'une même solution ne sera pas visitée plusieurs fois. Leur efficacité est donc empirique et dépend de l'instance considérée. Toutefois, dans la pratique, ils fournissent souvent de très bons résultats en un temps raisonnable pour une grande variété de problèmes. De plus, ils ont l'avantage de pouvoir résoudre des problèmes sur-contraints car ils manipulent des solutions éventuellement non-consistantes. Ils peuvent également fournir une solution à tout moment (algorithme *anytime*) car l'état courant est une instanciation totale.

5.6.3 Concepts communs aux métaheuristiques de RL

Nous abordons ici les concepts qui sont communs aux différents algorithmes de recherche locale.

5.6.3.1 Solution initiale

Il existe deux stratégies principales pour générer une solution initiale : l'approche aléatoire et l'approche gloutonne.

La génération aléatoire d'une solution initiale est rapide. Cette stratégie peut entraîner une déviation importante en termes de solutions obtenues et constitue à ce titre une bonne base de diversification. Néanmoins, pour certains problèmes contraints, il peut être difficile de générer des solutions faisables avec cette technique. Dans ce cas, les algorithmes de construction gloutons représentent une bonne alternative. Généralement, une heuristique gloutonne peut être utilisée avec une complexité réduite à un temps polynomial. Par rapport aux algorithmes aléatoires, les algorithmes gloutons conduisent souvent à une solution initiale

de meilleur qualité. Dans ce cas, la RL nécessite généralement moins d'itérations pour converger vers un optimum local. Mais rien ne garantit que l'utilisation d'une solution initiale de qualité supérieure conduira systématiquement la RL à un meilleur optimum local. Habituellement, on constate que plus le voisinage est grand, moins les performances de la RL sont sensibles à la solution initiale [266].

Des règles métier identifiées par des experts ou des solutions déjà implémentées peuvent également servir de support pour générer une solution initiale.

5.6.3.2 Voisinage

La définition d'un voisinage est une opération délicate qui est requise pour utiliser n'importe quelle métaheuristique de RL. Sa structure joue un rôle essentiel sur les performances. De plus, si elle n'est pas adéquate au problème donné, la résolution échouera quelle que soit la méthode employée. La modélisation du problème d'optimisation et le choix du voisinage doivent être effectués de telle sorte qu'il existe au moins un chemin entre chaque solution $s' \in \mathcal{S}$ menant à une solution optimale s^*.

Definition 10 (Voisinage)**.** Étant donné une solution s, le voisinage de s est un sous-ensemble de solutions noté $\mathcal{N}(s) \subseteq \mathcal{S}$; $\mathcal{N}(s)$ contenant des solutions directement atteignables à partir d'une transformation élémentaire donnée de s. Une solution $s' \in \mathcal{N}(s)$ est dite voisine de s.

Un voisin est généré par l'application d'une transformation élémentaire qui réalise une légère perturbation de la solution s. Par exemple, une transformation élémentaire peut consister à effectuer une permutation de valeurs entre deux variables (2-échanges) d'une même solution s. Dans ce cas, la taille du voisinage $\mathcal{N}(s)$ est égale à $\frac{n(n-1)}{2}$, avec n le nombre de variables du problème.

La structure du voisinage dépend de la transformation autorisée. Ainsi, à partir d'une solution donnée, on peut établir plusieurs structures de voisinage différentes selon les transformations retenues.

Definition 11 (Optimum local)**.** Une solution s est un optimum local si et seulement si il n'existe pas une autre solution $s' \in \mathcal{N}(s)$ avec un coût (issu de la fonction objectif) strictement meilleur.

$$\forall s' \in \mathcal{N}(s) \begin{cases} f(s) \leq f(s') & \text{pour un problème de minimisation} \\ f(s) \geq f(s') & \text{pour un problème de maximisation} \end{cases}$$

La taille du voisinage pour une solution s correspond au nombre de voisins de s. Utiliser de grands voisinages peut améliorer la qualité des solutions obtenues dans la mesure où on considère plus de voisins à chaque itération. Mais dans ce cas, un temps de calcul supplémentaire sera requis pour effectuer chaque itération. Lorsque le nombre de cycles à opérer est important pour la résolution d'un problème, le temps additionnel occasionné peut devenir conséquent. Il faut donc trouver un compromis entre la qualité du voisinage et la complexité nécessaire pour l'explorer. Lorsque la complexité d'exploration d'un voisinage devient trop importante, ce problème devient à lui seul un problème d'optimisation. Il est alors indispensable de concevoir des procédures efficaces pour réaliser l'exploration. En général, ces procédures cherchent à identifier les meilleurs voisins (ou des voisins meilleurs) sans devoir énumérer l'ensemble du voisinage [5].

5.6.3.3 Paysage d'un problème d'optimisation

Le paysage d'un problème d'optimisation combinatoire est défini par le triplet $(\mathcal{S}, \mathcal{N}, f)$ tel que :

- $\mathcal{S}$: est l'ensemble des solutions réalisables ;
- $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$: est une relation de voisinage qui associe à toute solution s de l'ensemble des solutions un sous-ensemble $\mathcal{N}(s)$ de solutions réalisables appelées voisins ;
- $f : \mathcal{S} \to \mathbb{R}$: est une fonction objectif qui mesure la qualité des solutions réalisables.

La notion de paysage a été introduite pour la première fois dans la littérature par WRIGHT [289] en lien avec la biologie. La structure de voisinage a été utilisée pour étudier les dynamiques des systèmes

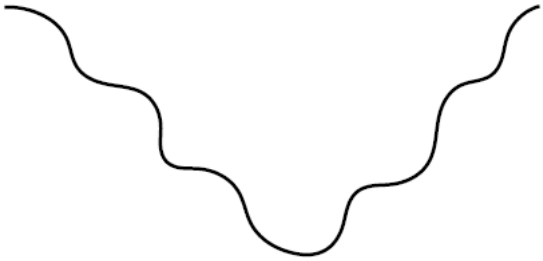
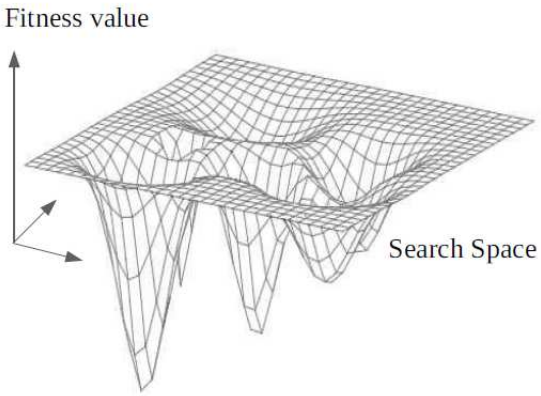


FIGURE 5.5 – Paysage lisse d'après [181].

FIGURE 5.6 – Paysage rugueux d'après [181].

FIGURE 5.7 – Paysage d'un problème d'optimisation d'après [181].



biologiques et comprendre les processus d'évolution déterminés par des opérateurs comme la mutation ou le croisement. Cette notion fut ensuite transposée à d'autres domaines comme celui de l'optimisation combinatoire [257, 256].

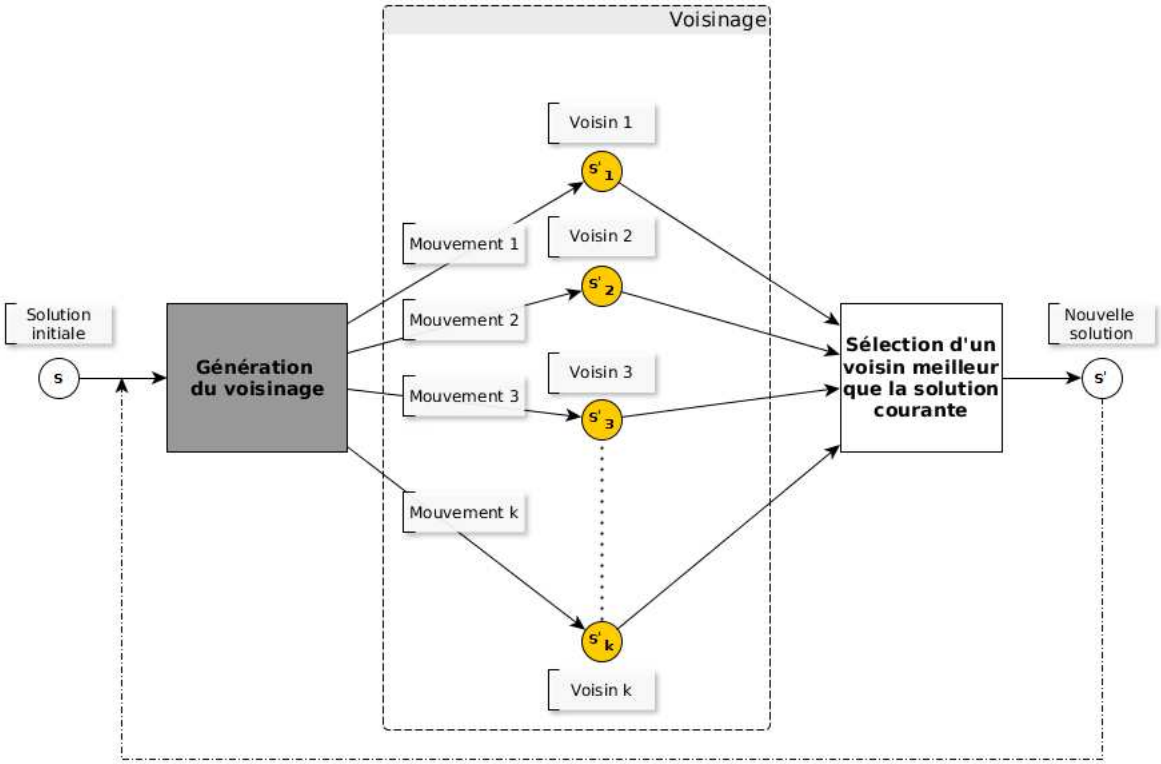
On peut utiliser des termes géographiques pour décrire un paysage : l'espace de recherche représente le sol, la relation de voisinage connecte les solutions les unes aux autres et la fonction objectif leur affecte une qualité transcrite en altitude (figure 5.7). On obtient un paysage constitué de vallées, de plaines, de sommets, de plateaux, de cuvettes, etc. Analyser la structure d'un paysage permet d'appréhender le degré de difficulté d'un problème et peut aider à concevoir de meilleures représentations du voisinage. Il est plus facile d'atteindre l'optimum global pour un paysage lisse (figure 5.5) que pour un paysage rugueux (figure 5.6) où les optima locaux sont plus nombreux : on peut alors expliquer la réussite ou l'échec d'une métaheuristique sur un problème donné.

Globalement, un paysage peut avoir un relief plat, rugueux ou vallonné [181]. Beaucoup de problèmes d'optimisation sont caractérisés par un paysage plat dans lequel il y a beaucoup de plateaux, c'est à dire beaucoup de voisins ayant la même qualité [266]. Lorsqu'une métaheuristique rencontre un plateau, aucune information ne lui permet de guider sa recherche. Pour casser ces plateaux, une stratégie consiste à intégrer dans la fonction objectif des informations supplémentaires permettant de discriminer les solutions ayant la même valeur d'objectif [132, 84].

5.6.3.4 Évaluation incrémentale du voisinage

L'évaluation de la fonction objectif peut nécessiter un temps de calcul important. Une exploration naïve du voisinage d'une solution s consiste à évaluer complètement la fonction objectif pour chaque voisin s' de s. Lorsque cela est possible, il est plus performant d'évaluer chaque voisin par une évaluation incrémentale représentée par $\triangle(s, m)$ où s est la solution courante et m le mouvement (ou transformation) appliqué à s pour obtenir un voisin s'. On limite ainsi l'évaluation à la transformation de s au lieu d'évaluer complète-

FIGURE 5.8 – Principe de génération du voisinage à partir de la solution courante et sélection d'un voisin d'après [35].



ment la solution voisine s'. Cette évaluation incrémentale peut être simple ou très complexe à implémenter en fonction du problème et de la structure du voisinage.

5.6.4 Méthodes de descente (Simple Improvement-based Local Search)

Les méthodes de descente (amélioration continue) n'utilisent pas de stratégies avancées et à ce titre nous les considérons comme des techniques de base pouvant être utilisées par différentes métaheuristiques plus élaborées. On les retrouve sous le nom "*hill climbing*" pour les problèmes de maximisation.

Une *méthode de descente* (*first improvement* ou *simple hill climbing*) choisit à chaque étape une solution voisine meilleure que la solution courante (figure 5.8). Quand plus aucune solution voisine n'améliore la solution courante, un optimum local est atteint et le processus s'arrête. La *méthode de plus forte descente* (*best improvement* ou *steepest ascent hill climbing*) est une variante qui choisit à chaque étape la meilleure solution voisine parmi celles qui améliorent la solution courante.

En termes d'exploration du voisinage, les méthodes de descente (first improvement) font une exploration partielle du voisinage (algorithme 4) alors que les méthodes de plus forte descente (best improvement) nécessitent une exploration exhaustive du voisinage (algorithme 5).

L'avantage principal de ces méthodes de descente réside dans leur grande simplicité et leur rapidité. Historiquement, ces méthodes ont toujours compté parmi les méthodes heuristiques les plus populaires pour traiter les problèmes d'optimisation combinatoire. Toutefois, elles comportent deux obstacles majeurs qui peuvent limiter considérablement leur efficacité :

- suivant la taille et la structure du voisinage $\mathcal{N}$ considéré, la recherche de la meilleure solution voisine est un problème qui peut être aussi difficile que le problème initial.
- une méthode de descente est incapable de progresser au-delà du premier optimum local rencontré.

Algorithme 4 : Méthode de descente

```
Créer la solution initiale s ;
while (optimum local non atteint) do
    repeat
        Créer un nouveau voisin s' en réalisant un mouvement sur s ;
    until (s' meilleur que s) ou (plus aucun mouvement disponible) ;
    if (s' meilleur que s) then
        Passer à la solution s' (la solution courante s est remplacée par s') ;
return s ;
```

Algorithme 5 : Méthode de plus forte descente

```
Créer la solution initiale s ;
while (optimum local non atteint) do
    Déterminer le voisinage complet 𝒩 de la solution courante s ;
    if (𝒩 contient au moins une meilleure solution que s) then
        Choisir la meilleure solution s' dans 𝒩 ;
        Passer à la solution s' (la solution courante s est remplacée par s') ;
return s ;
```

Or, les problèmes d'optimisation combinatoire comportent typiquement de nombreux optima locaux pour lesquels la valeur de la fonction objectif peut être fort éloignée de la valeur optimale.

Pour remédier à ce dernier problème, la solution la plus simple est la *méthode de descente avec relance aléatoire* (*random restart hill climbing*) qui consiste à générer une nouvelle solution de départ de façon aléatoire et à relancer la méthode de descente. On remarquera cependant que cette solution ne tire aucun profit des optima locaux déjà découverts. Une autre solution consiste à accepter des voisins de même qualité que la configuration courante. Cette approche permet à la recherche de se déplacer sur les plateaux, mais n'est pas suffisante pour ressortir d'un optimum local.

L'efficacité des méthodes de RL repose en réalité sur un équilibre entre deux notions essentielles représentées par l'intensification et la diversification.

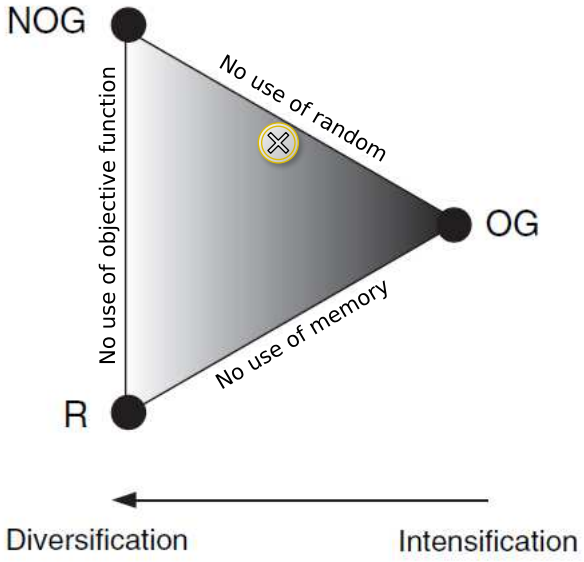
5.6.5 Mécanismes d'intensification et de diversification

Ces deux termes ont été initialement introduits par GLOVER dans le contexte de la recherche tabou [112, 113, 115] avant d'être généralisés puis vus comme des propriétés essentielles des métaheuristiques. L'*intensification* (ou exploitation) insiste sur l'examen en profondeur d'une zone de recherche particulière alors que la *diversification* (ou exploration) met en avant la capacité de découvrir des zones de recherche prometteuses.

L'idée de base de l'intensification est d'exploiter une ou des solutions prometteuses. Elle se fonde sur les notions liées à l'apprentissage en considérant que les attributs souvent présents dans les configurations d'élite connues sont favorables, il faut donc les exploiter plus que les autres. Par contre, les attributs rarement présents dans les configurations d'élite connues sont peu favorables, il faut donc les éviter. De nombreuses techniques d'intensification ont été proposées [161, 30, 231, 128] :

- relancer la recherche à partir des bonnes solutions déjà rencontrées ;
- reconstruire une solution de départ qui tente de combiner des attributs souvent rencontrés dans les meilleures configurations ;

FIGURE 5.9 – Classification (I&D) des composants d'une stratégie de recherche proposée par BLUM et ROLI [30] (**OG** = composants exclusivement guidés par la fonction objectif ; **NOG** = composants I&D uniquement guidés par une ou plusieurs fonctions autres que la fonction objectif ; **R** = composants guidés de façon totalement aléatoire). Le composant repéré par une croix 'X' situé à mi-chemin entre l'intensification et la diversification n'est pas stochastique.



- fixer certains attributs qui ont souvent été présents dans les configurations d'élite.

Cependant, l'application du seul principe d'intensification ne permet pas une recherche efficace. En effet, elle conduit à confiner la recherche dans une zone limitée qui finit par s'épuiser. Le cas de l'amélioration continue illustrée par les méthodes de descente (voir la section 5.6.4) rapidement piégées dans un optimum local représente bien ce phénomène.

A contrario, la diversification est un mécanisme qui oblige la recherche à s'éloigner d'une zone où les solutions sont connues pour se diriger vers d'autres zones inexplorées. HOOS et STÜTZLE ont établi un lien étroit entre diversification et caractère aléatoire [135]. Pour eux, une part d'aléatoire introduite dans la phase d'initialisation (pour créer la solution initiale) ou dans les étapes de recherche (en autorisant parfois la dégradation du score) permet de contourner les principales limitations des méthodes de descente qui restent piégées dans le premier minimum local rencontré.

De nombreuses techniques de diversification ont été proposées. La technique la plus simple consiste à construire une nouvelle solution de départ pour relancer la recherche. Les changements de voisinage (5.6.10.2) sont aussi une forme de diversification permettant d'atteindre de nouvelles solutions. D'autres heuristiques peuvent réaliser une perturbation aléatoire (modifications aléatoires de la solution courante) ou encore caractériser les régions visitées pour pouvoir ensuite s'en éloigner.

La recombinaison constitue un autre principe général qui complète l'intensification et la diversification. Elle consiste à construire de nouvelles configurations en combinant la structure de deux ou plusieurs bonnes configurations déjà trouvées. Cette idée issue des algorithmes génétiques (le croisement) a été introduite dans des méthodes de voisinage, notamment dans la méthode de recherche par dispersion (Scatter Search).

Une stratégie de recherche efficace doit donc intégrer à la fois des mécanismes d'intensification et de diversification et trouver un bon équilibre entre ces deux notions antagonistes. Bien que certains composants d'une stratégie de recherche ne puissent être clairement catalogués comme participant soit à l'intensification, soit à la diversification (car ils participent aux deux à la fois), BLUM et ROLI ont proposé une classification [30] représentée par un triangle dans lequel on place les composants pour les évaluer en

fonction de leur position à l'intérieur du triangle (figure 5.9). Dans ce triangle, l'effet de l'intensification diminue lorsque la distance au coin **OG** augmente tandis que la force de diversification augmente lorsqu'on s'approche des coins **R** ou **NOG**. Ce schéma correspond à une signature simple du composant concerné (e.g. composant repéré par une croix 'X' dans la figure 5.9) et permet ainsi de représenter les mécanismes qu'il met en œuvre en relation avec sa position.

5.6.6 Recherche Locale Itérée (Iterated Local Search, ILS)

La méthode de *recherche locale itérée* est une métaheuristique simple mais puissante [260, 176, 177, 184]. Il s'agit d'une amélioration des méthodes de descente qui propose une stratégie pour pallier l'arrêt de la recherche sur le premier optimum local rencontré, problème inhérent aux techniques de descente. A partir d'une solution initiale soumise à une méthode de descente classique, cette méthode enchaîne les trois étapes suivantes jusqu'à ce qu'un critère d'arrêt soit satisfait :

- perturbation aléatoire de la solution courante ;
- optimisation de la solution perturbée par une méthode de descente jusqu'à obtenir un optimum local ;
- acceptation (ou non) du nouvel optimum local comme solution courante.

La phase de perturbation est essentielle et elle doit être bien dosée. En effet, si la perturbation est trop faible, on ne pourra pas s'échapper du bassin d'attraction de l'optimum local précédent. Au contrainte, si la perturbation est trop importante, l'algorithme se comportera comme une méthode de descente avec relance aléatoire.

Pour ce qui est de l'acceptation d'une nouvelle solution courante, on peut imaginer un grand nombre de possibilités comprises entre deux extrêmes qui consistent à (i) accepter le nouvel optimum local uniquement si il améliore la dernière solution trouvée ou à (ii) accepter systématiquement la nouvelle solution.

Algorithme 6 : Méthode de recherche locale itérée

```
1 Créer la solution initiale s_0 ;
2 s* = Méthode de descente(s_0) ;
3 repeat
4     s_1 = Perturbation aléatoire(s*, historique) ;
5     s_2 = Méthode de descente(s_1) ;
6     if (Acceptation(s*, s_2, historique)) then
7         s* = s_2 ;
  until (critère de fin atteint) ;
```

5.6.7 Recuit Simulé (Simulated Annealing, SA)

Le recuit simulé est une métaheuristique populaire dont le mécanisme de recherche est calqué sur l'algorithme de Metropolis [189] et les principes de recuit thermodynamiques utilisés en métallurgie. KIRKPATRICK et al. [153] et CERNY [42] ont été les premiers à s'inspirer d'une telle technique pour résoudre des problèmes d'optimisation combinatoire.

En partant d'une température élevée où le métal est liquide, on le refroidit progressivement en tentant de trouver le meilleur équilibre thermodynamique. Chaque niveau de température est maintenu jusqu'à obtenir un équilibre. Dans ces phases de température constante, on peut passer par des états intermédiaires du métal non satisfaisants, mais conduisant à la longue à des états meilleurs. La vitesse de refroidissement a un impact crucial sur la configuration finale, un alignement atomique approprié n'étant possible que si le procédé de refroidissement est suffisamment lent.

Le voisinage $\mathcal{N}(s)$ d'une solution $s \in \mathcal{S}$ s'apparente à l'ensemble des états atteignables depuis l'état courant en faisant subir des déplacements infinitésimaux aux atomes d'un système physique. A chaque

itération de l'algorithme, une seule solution voisine s' est générée à partir du voisinage $\mathcal{N}(s)$. Celle-ci est acceptée si elle est meilleure que la solution courante s. Dans le cas contraire, la solution s' est acceptée avec une certaine probabilité $prob(\triangle f, T)$ qui dépend de la détérioration de la qualité $\triangle f = f(s') - f(s)$ et d'un paramètre T correspondant à la température.

Les changements de température sont effectués sur la base d'un schéma de refroidissement précis. En règle générale, la température est diminuée par paliers à chaque fois qu'un certain nombre d'itérations est effectué. L'algorithme est interrompu lorsqu'aucune solution voisine n'a été acceptée pendant un cycle complet d'itérations à température constante. La performance du recuit simulé est étroitement liée au schéma de refroidissement considéré.

De nombreuses études théoriques ont été effectuées à ce sujet et plusieurs variantes ont été proposées [52, 211].

Algorithme 7 : Méthode de recuit simulé

```
Créer la solution initiale s ;
Définir la température initiale T_0 ;
Définir le nombre d'essais pour chaque niveau de température L ;
Initialiser le compteur de niveau k ← 0 ;
while (critère de fin non satisfait) do
    for (i = 1 to L) do
        Créer un nouveau voisin s' en appliquant à s un mouvement choisi aléatoirement ;
        Calculer la différence de coût △f = f(s') − f(s) ;
        if (△f ≤ 0) then
            Passer à la solution s' (Remplacer la solution courante s par s') ;
        else
            Tirer un nombre aléatoire r ∈ [0, 1] ;
            if ( r ≤ exp( −△f / T_k ) ) then
                Passer à la solution s' (Remplacer la solution courante s par s') ;
    Actualiser la meilleure solution trouvée (si nécessaire) ;
    Définir k ← k + 1 ;
    Définir la valeur de la température T_k pour le nouveau niveau k ;
return la meilleure solution trouvée ;
```

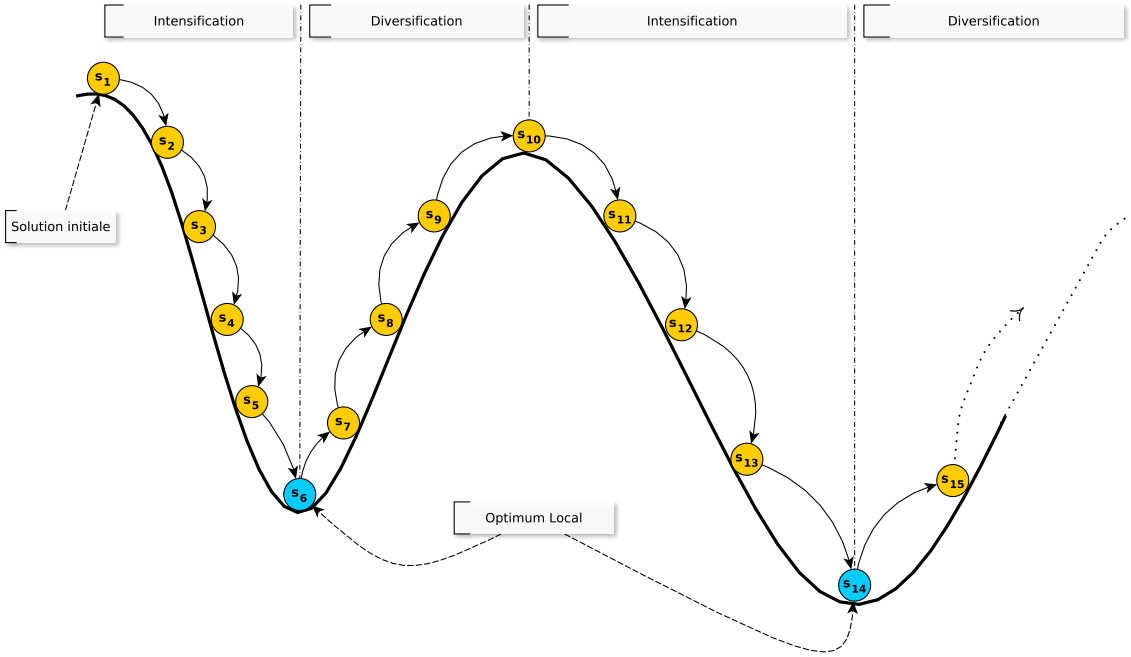
En début du processus de recherche, le système est chauffé en fixant T à une valeur élevée. L'algorithme de Metropolis est exécuté pour un nombre déterminé d'itérations, tout en gardant la température fixe. Après cela, la température diminue lentement et l'algorithme de Metropolis est relancé. Ce plan d'action est répété jusqu'à ce que le système soit gelé, c'est à dire lorsqu'on a atteint une température très basse et que pratiquement aucune modification de configuration ne se produit (algorithme 7).

5.6.8 Recherche Tabou (Tabu Search, TS)

Introduite indépendamment par F. GLOVER [111] d'une part et par P. HANSEN [125] d'autre part, la recherche tabou est une métaheuristique qui construit un voisinage en excluant un certain nombre de configurations récemment visitées (ou mouvements récents). Elle utilise toujours la meilleure solution voisine indépendamment du fait qu'elle améliore ou non la qualité de la solution précédente.

Tant qu'on ne se trouve pas dans un optimum local, cette méthode se comporte comme une méthode de plus forte descente qui améliore à chaque itération la valeur de la fonction objectif en choisissant une configuration meilleure. Mais lorsqu'elle a atteint un optimum local, la recherche tabou cherche à s'en

FIGURE 5.10 – Recherche tabou pour un problème de minimisation : déplacement dans l'espace de recherche en sélectionnant à chaque itération le meilleur voisin dans un voisinage débarrassé des solutions récentes marquées tabou (pour éviter les cycles). On alterne ainsi entre des phases d'intensification et de diversification qui poussent la recherche à se diriger vers un optimum global.



échapper en choisissant le voisin le moins mauvais, c'est à dire celui qui détériore le moins le coût de la dernière solution sélectionnée (figure 5.10).

Pour éviter d'emprunter le chemin inverse (retour à une solution précédente) qui provoquerait un cycle, la méthode se déplace d'une configuration à une autre en s'interdisant de revenir sur une configuration récente déjà rencontrée. Elle utilise pour cela une liste tabou qui contient les dernières configurations visitées et interdites pendant un certain nombre d'itérations, le but étant de donner assez de temps à l'algorithme pour lui permettre de sortir du minimum local.

Cependant, stocker dans la liste tabou des configurations entières et vérifier si elles correspondent à la prochaine destination peut être coûteux en espace mémoire et en temps de calcul. Pour éviter ce problème, une alternative courante consiste à stocker et à interdire un ensemble de mouvements récemment utilisés (plutôt que des solutions), ces mouvements pouvant ramener à une configuration déjà visitée.

Dans le cas d'une liste tabou basée mouvements, l'interdiction d'emprunter un chemin ne porte pas uniquement sur les solutions récemment visitées, elle porte plus globalement sur toutes les solutions qui résulteraient de l'un des mouvements présents dans la liste. Le nombre de ces solutions involontairement déclarées tabou dépend de la capacité de stockage de la liste tabou et peut varier très fortement en fonction du problème et de la structure du voisinage. Il arrive même que cette interdiction empêche de visiter des solutions supérieures (à la meilleure configuration déjà rencontrée) n'ayant pas encore été visitées, ce qui rend le processus de recherche plus difficile. Pour ne pas interdire l'accès à ces configurations de qualité supérieure, on peut ajouter un mécanisme dit d'*aspiration* qui permet d'enlever le caractère tabou à certains mouvements jugés utiles pour accéder à des solutions de qualité exceptionnelle. D'autres critères d'aspiration plus sophistiqués ont également été proposés [66, 131].

La durée tabou est un paramètre essentiel qui influence énormément les performances de la méthode, il est donc crucial de bien la choisir en fonction du problème traité et de l'instance. La durée du statut tabou peut être soit statique soit dynamique. Une durée statique est fixée en début de résolution et reste constante durant toute la recherche. Une durée dynamique peut varier au cours de l'exécution. Plusieurs

études [127] ont montré que les résultats obtenus avec les listes tabou dynamiques pouvaient être meilleurs qu'avec une liste statique. Dans [72, 73], les auteurs proposent des techniques d'apprentissage pour définir empiriquement les paramètres de durée.

Algorithme 8 : Méthode de recherche tabou

```
1 Créer la solution initiale s ;
2 Initialiser la liste tabou T ;
3 while (critère de fin non satisfait) do
4     Déterminer le voisinage complet 𝒩 de la solution courante s ;
5     Choisir la meilleure solution s' non tabou dans 𝒩 ;
6     Passer à la solution s' (la solution courante s est remplacée par s') ;
7     Modifier la liste tabou T ;
8     Actualiser la meilleure solution trouvée (si nécessaire) ;
9 return la meilleure solution trouvée ;
```

5.6.8.1 La liste tabou

La liste tabou contient habituellement une quantité limitée d'informations pouvant être de différentes natures [106]. Elle est généralement organisée comme un buffer de type "*first in first out*". Lorsque le buffer est plein et qu'un nouvel élément doit y être ajouté, l'élément le plus ancien est supprimé du buffer et les éléments précédents sont décalés vers la fin. La capacité du buffer détermine la "durée tabou". Notons qu'une liste tabou de taille fixe ne peut pas éviter avec certitude l'occurrence de cycles. Afin d'améliorer sa protection contre les cycles, deux approches majeures ont été développées :

- faire varier la taille de la liste au fil du temps [112, 113, 250, 263].
- faire varier la durée tabou de chaque élément [107]. Cette approche nécessite une structure mémoire adaptée. Les éléments tabou doivent être mémorisés dans un tableau avec une information de durée tabou, typiquement le numéro d'itération à atteindre pour qu'ils passent du statut tabou au statut non tabou.

5.6.8.2 Alternance entre intensification et diversification

Afin d'illustrer la stratégie de recherche tabou et le nécessaire équilibre entre intensification et diversification, revenons sur la Figure 5.10.

Le processus d'optimisation démarre avec la solution initiale s_1. Il entame alors une première phase d'intensification de la recherche en opérant une méthode de descente jusqu'à ce qu'il rencontre un optimum local (s_6). Pour sortir de cette impasse, il a alors deux solutions : soit il revient à la solution s_5, soit il avance à la solution s_7. La liste tabou ayant mémorisé les dernières solutions visitées, on suppose que la solution s_5 est interdite. La recherche tabou passe donc à la solution s_7 correspondant à la meilleure solution voisine non tabou. La solution s_7 ayant été visitée, elle devient à son tour tabou. La recherche n'a pas d'autre choix que de poursuivre sa phase de remontée jusqu'à atteindre la solution s_{10}. Cette ascension l'éloigne peu à peu de la vallée où la recherche était bloquée : il s'agit d'une phase de diversification qui guide la recherche vers de nouveaux horizons. A partir du point s_{10}, des solutions voisines d'amélioration non tabou sont accessibles et le processus rentre à nouveau dans une phase d'intensification.

5.6.9 Adaptive Search (AS)

Adaptive Search est une méthode de recherche locale basée contraintes (i.e. Constraint-Based Local Search) proposée au début des années 2000 [47, 48]. Cette métaheuristique tire parti de la structure d'un problème

en exploitant les contraintes et les variables d'un CSP donné. Elle peut ainsi guider la recherche avec plus de précision qu'une simple fonction de coût globale à optimiser, qui mentionnerait uniquement le nombre de contraintes violées [10]. L'algorithme utilise également une mémoire à court terme, dans l'esprit de la recherche tabou, pour éviter (i) une stagnation dans un optimum local et (ii) des cycles infinis. De part sa nature, cette métaheuristique gère de façon intrinsèque les problèmes sur-contraints.

L'algorithme 9 présente un schéma général de la métaheuristique Adaptive Search qui s'articule autour de trois principes :

- considérer pour chaque contrainte une fonction heuristique capable de calculer un degré de satisfaction des objectifs : l'erreur courante sur la contrainte ;
- regrouper les contraintes sur chaque variable et projeter les erreurs sur les variables dans le but de réparer la plus mauvaise variable avec la valeur la plus prometteuse ;
- conserver dans une mémoire à court terme les mauvaises configurations pour éviter les cycles (i.e. à la manière d'une liste tabou) et proposer un mécanisme de réinitialisation partielle.

Pour chaque contrainte, une fonction d'erreur doit être définie afin de fournir, pour chaque tuple "variable | valeur", une indication sur son degré de violation. Cette idée a également été proposée indépendamment par P. GALINIER et J.K. HAO [99], où il est question de "fonctions de pénalité", puis réutilisée dans la plateforme *Comet* [272], où les auteurs parlent de "violations".

AS procède à une réparation itérative sur la base des erreurs liées aux variables et aux contraintes, en cherchant à réduire l'erreur sur la variable ayant le coût le plus élevé. Pour cela, on calcule la fonction d'erreur pour chaque contrainte, puis on combine pour chaque variable les erreurs de toutes les contraintes où elles apparaissent, ce qui permet de projeter les erreurs des contraintes sur les variables correspondantes. La procédure qui permet de transférer les erreurs des contraintes sur les variables est problème-dépendant. Il s'agit généralement d'une simple somme, ou d'une somme de valeurs absolues, bien que cela puisse également être une somme pondérée si les contraintes sont données avec des priorités différentes. Finalement, la variable avec l'erreur la plus importante est désignée comme étant "la coupable" et sa valeur est modifiée. Pour cette seconde étape, l'heuristique *min-conflicts*[11] [192] est utilisée pour sélectionner la valeur dans le domaine de la variable qui est la plus prometteuse, c'est à dire la valeur pour laquelle l'erreur totale dans la configuration suivante est minimale.

Pour éviter d'être piégée dans un minimum local, la méthode AS incorpore un mécanisme de mémoire à court terme permettant de mémoriser les configurations à éviter (les variables peuvent être marquées tabou et être gelées pour un nombre d'itérations). Elle intègre également des étapes de réinitialisation. Une réinitialisation consiste à assigner de nouvelles valeurs aléatoires pour certaines variables, ces dernières étant également choisies de façon aléatoire. Par exemple, lorsque trop de variables deviennent tabou, l'algorithme risque d'être piégé autour d'un minimum local. Une diversification peut alors être sollicitée par une réinitialisation partielle en assignant de nouvelles valeurs à un pourcentage donné de variables du problème, les variables et les valeurs étant choisies aléatoirement. Une réinitialisation peut être déclenchée lors d'une itération dès qu'un nombre de variables marquées tabou est atteint.

Par rapport à l'algorithme de base (cf. algorithme 9), une amélioration peut être apportée lorsqu'on se déplace sur un plateau. Dans ce cas, un système stochastique simple peut être utilisé pour décider de poursuivre sur le plateau ou de s'en échapper, selon une certaine probabilité.

Bien que cette méthode soit très simple, elle n'en reste pas moins très efficace pour résoudre des problèmes combinatoires complexes [49]. Soulignons également que cette métaheuristique exploite plusieurs mécanismes stochastiques. D'une part dans l'algorithme de base pour la sélection d'une variable et d'une valeur pour trancher entre des choix équivalents. D'autre part, dans la valeur des paramètres de contrôle pouvant être fixés différemment par l'utilisateur à chaque exécution (% de variables concernées par une ré-

[11] La méthode *min-conflicts* cherche à éviter une exploration exhaustive du voisinage à chaque itération comme le fait *la méthode de plus forte descente* (ou *best improvement*). Pour ce faire, une variable en conflit (impliquée dans une contrainte violée) est choisie aléatoirement et instanciée avec la valeur qui minimise le nombre de conflits, avec un choix aléatoire en cas d'égalité. Cette heuristique introduite pour des problèmes de satisfaction de contraintes tente ainsi de réparer une configuration non consistante.

Algorithme 9 : Algorithme de base pour la métaheuristique Adaptive Search, d'après [10].

Input : problème donné au format CSP :
- variables X_i avec leurs domaines
- contraintes C_j avec fonctions d'erreur
- fonction pour projecter les erreurs sur les variables
- fonction de coût à minimiser

certains paramètres de règlage :
- *TT* : # iterations où une variable reste tabou
- *RL* : # variables tabou pour déclencher un reset
- *RP* : % de variables à réinitialiser
- *MI* : # max. iterations avant un redémarrage
- *MR* : # max. de redémarrages

Output : une solution si le CSP est satisfait ou une solution approchée de coût minimal sinon.

```
1  Restart ← 0 ;
2  Opt_Sol ← ∅ ;
3  Opt_Cost ← MAX ;
4  repeat
5      Restart ← Restart + 1 ;
6      Iteration ← 0 ;
7      Calculer une affectation aléatoire A de variables dans V ;
8      if (cost(A) < Opt_Cost) then
9          Opt_Sol ← A ;
10         Opt_Cost ← cost(A) ;
11     repeat
12         Iteration ← Iteration + 1 ;
13         Calculer les erreurs de toutes les contraintes dans C et combiner les erreurs sur chaque variable;
               (en considérant uniquement les contraintes où la variable apparaît)
14         Sélectionner la variable X (non marquée tabou) avec l'erreur la plus forte ;
               (si il y en a plusieurs avec le même coût, en sélectionner une de façon aléatoire dans la liste)
15         Évaluer les configurations du voisinage de A à partir de X ;
16         if (aucun changement ne permet d'améliorer le coût) then
17             marquer X comme tabou jusqu'à l'itération numéro : Iteration + TT ;
18             if (le nombre de variables marquées tabou ≥ RL) then
19                 réinitialiser au hasard RP % variables dans V (et ne plus les considérer comme tabou) ;
           else
20             Appliquer le meilleur changement à partir de X, donnant la configuration suivante A' ;
21             A ← A' ;
22             if (cost(A) < Opt_Cost) then
23                 Opt_Sol ← A ;
24                 Opt_Cost ← cost(A) ;
       until (Opt_Cost = 0 (une solution est trouvée) ou Iteration ≥ MI) ;
   until (Opt_Cost = 0 (une solution est trouvée) ou Restart ≥ MR) ;
25 output(Opt_Sol, Opt_Cost)
```

initialisation partielle, nombre maximum de variables marquées tabou pour déclencher une réinitialisation partielle, ...).

5.6.10 Autres métaheuristiques obtenues par recombinaison de méthodes simples

Nous présentons ici quelques méthodes de RL originales qui combinent et exploitent des techniques de base déjà abordées dans ce chapitre. L'association de techniques simples pour produire des techniques plus sophistiquées nous permet d'illustrer tout le potentiel que représente l'hybridation de méthodes (complètes et incomplètes) sans toutefois développer ce dernier aspect [127, 252, 162, 146, 76, 152].

5.6.10.1 Greedy Randomized Adaptive Search Procedure (GRASP)

La méthode GRASP [91, 230] consiste à répéter deux phases successives tant qu'un critère d'arrêt n'est pas satisfait :

- la première phase construit une solution s par un algorithme glouton aléatoire (Greedy randomized, section 5.5.2) ;
- la seconde phase utilise une méthode de descente pour trouver un optimum local à partir de s.

La méthode retourne la meilleure solution trouvée.

5.6.10.2 Recherche à voisinage variable (Variable Neighborhood Search, VNS)

Proposée par HANSEN et MLADENOVIC [126] en 1997, la méthode de recherche à voisinage variable est une métaheuristique récente qui exploite un changement de structures de voisinage pour s'échapper d'un optimum local. L'idée de base est de faire varier la structure de voisinage en se basant sur le fait qu'une configuration s reconnue comme un optimum local pour une structure de voisinage donnée peut ne pas être un optimum local pour une autre structure de voisinage.

Cette méthode fonctionne en plusieurs étapes. Tout d'abord, elle nécessite la définition de différentes structures de voisinage, notées $\mathcal{N}_1$, ..., $\mathcal{N}_{k_{max}}$, allant d'une portée faible à une portée de plus en plus importante. A partir d'une configuration complète de départ s, une solution s' voisine de s est sélectionnée (de façon aléatoire) en utilisant la structure de voisinage $\mathcal{N}_1$ (de plus faible portée). On applique alors une méthode de descente sur s'. Si la solution s'' obtenue à l'issue de cette première recherche est meilleure que la solution s de départ alors la solution s est remplacée par s'' et la recherche reprend à partir de la première structure de voisinage. Sinon, une autre configuration complète s' peut être sélectionnée dans le voisinage de s en utilisant une structure de voisinage de portée supérieure (soit dans notre cas la structure de voisinage $\mathcal{N}_2$). La même règle est appliquée pour les choix suivants jusqu'à l'obtention d'un critère d'arrêt. A noter que les règles de changement de structure de voisinage peuvent être plus complexes.

Il peut être intéressant d'utiliser des algorithmes de ce type pour des problèmes disposant de nombreux minimums locaux.

5.6.10.3 Recherche Locale Guidée (Guided Local Search, GLS)

La recherche locale guidée proposée par VOUDOURIS et TSANG [281] pénalise les propriétés caractéristiques des solutions qui apparaissent dans un optimum local. Elle consiste à modifier dynamiquement la fonction objectif pour exploiter ces pénalités et rendre moins attractifs les optima locaux.

La recherche démarre avec une solution initiale et toutes les valeurs de pénalité à zéro. Une méthode de descente est utilisée jusqu'à ce qu'un optimum local soit trouvé. La solution trouvée est utilisée pour calculer les nouvelles pénalités. Ces deux dernières étapes sont répétées jusqu'à ce qu'un critère d'arrêt soit vérifié.

5.6.11 Conclusion

Par rapport à cette étude relative aux méthodes de résolution incomplètes, nous souhaitons mettre en perspective deux aspects qui nous paraissent essentiels : des différences significatives entre les méthodes d'une part et des opportunités d'assemblage d'autre part.

D'un point de vue paramétrage, le recuit simulé paraît nettement plus compliqué que la méthode de recherche tabou par exemple : d'un côté, il faut définir un système de refroidissement par paliers (température initiale, taux de décroissance de la température, durée des paliers de température, ...) [181, 245] alors que d'un autre, il suffit de définir la taille d'une liste. Lorsqu'on doit satisfaire une contrainte de temps importante, la méthode de recuit simulé ne semble pas bien positionnée non plus avec un temps de calcul qui reste excessif dans certaines applications [245]. A l'inverse, les méthodes basées sur le principe de plus forte descente semblent intéressantes pour améliorer très fortement la qualité d'une solution en tout début du processus de résolution. On notera que la recherche tabou est une méthode aussi agressive qu'une méthode de descente mais, si le temps le permet, elle sera capable de dépasser le premier optimum local. Par rapport à un type de problème particulier, des instances de problèmes classiques sont disponibles pour pouvoir évaluer la pertinence d'une méthode et la comparer aux autres (cf. les librairies : OR-Library[12], TSPLIB[13], QAPLIB[14], ...). En fonction de la structure d'un problème ou d'une instance, une méthode sera donc plus ou moins adaptée et obtiendra parfois les meilleurs résultats connus pour certains problèmes. Il est donc important d'avoir une vision des différentes techniques disponibles pour retenir celle qui est la plus prometteuse. Par exemple, parmi les métaheuristiques basées sur une recherche locale, la méthode tabou semble être la plus performante pour résoudre les problèmes d'affectation quadratique. Pour ce probème et dans la majorité des cas, c'est elle qui présente en effet le meilleur compromis entre qualité des solutions et temps de calcul [187]. La méthode Adaptive Search quant à elle sait tirer parti de la structure d'un problème pour guider la recherche en exploitant ses variables et ses différentes contraintes lorsqu'elles sont exprimées sous la forme d'un CSP. Ces éléments confèrent à cette métaheuristique un intérêt particulier dans notre contexte de travail.

D'un autre coté, on s'aperçoit que chaque technique apporte son lot de contributions et qu'il est tout à fait possible de conjuguer certaines méthodes de base entre elles pour réaliser des méthodes plus sophistiquées. Nous retenons plus particulièrement les procédés suivants qui peuvent être complémentaires :

- partir de solutions initiales différentes construites de façon aléatoire ou par une méthode de construction gloutonne aléatoire ;
- enchaîner des phases successives constituées de différentes techniques et retourner le meilleur résultat trouvé ;
- exploiter l'agressivité des méthodes de plus forte descente ;
- réaliser des perturbations aléatoires sur les variables ;
- accepter des solutions moins bonnes ;
- exploiter une mémoire à court terme pour ne pas revenir sur ses pas ou ignorer certaines possibilités (configurations ou variables) ;
- utiliser la structure du problème pour guider la recherche en exploitant les contraintes et les variables d'un problème (défini par un CSP).

5.7 Métaheuristiques parallèles (à base de voisinage)

Les métaheuristiques offrent souvent la seule possibilité de résoudre des problèmes complexes du monde réel. Cependant, même en utilisant les métaheuristiques, les limites de ce qui peut être résolu dans un temps de calcul raisonnable sont encore trop rapidement atteintes pour beaucoup de problèmes. Dans sa

[12] http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[13] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
[14] http://anjos.mgi.polymtl.ca/qaplib/

thèse [102], Vincent GARDEUX parle de "*malédiction de la dimension*". D'autre part, les métaheuristiques n'offrent pas de garanties sur la qualité des solutions fournies et leurs performances dépendent souvent des caractéristiques particulières du problème ou de l'instance.

La parallélisation des métaheuristiques permet justement d'adresser ces deux problèmes : (i) offrir un maximum de performances et (ii) rendre les métaheuristiques robustes[15] [58]. Elle peut également être utilisée pour améliorer la qualité des solutions obtenues ou résoudre des problèmes de grande taille [266].

Mais en quoi consiste la parallélisation d'une métaheuristique ? Le calcul parallèle (ou distribué) met en œuvre différents processus qui travaillent simultanément sur différents processeurs pour résoudre une instance donnée d'un problème. Le parallélisme correspond à une décomposition de la charge totale de calcul et à la distribution des tâches correspondantes sur les processeurs disponibles. Concernant les métaheuristiques, il existe deux types majeurs de décomposition :

- décomposition algorithmique ;
- décomposition de l'espace de recherche ;

D'un point de vue algorithmique, la source principale de parallélisme pour les métaheuristiques concerne l'*évaluation concurrente des voisins*, c'est à dire l'exécution concurrente des boucles d'itération pour l'évaluation des voisins. Au niveau algorithmique, c'est souvent la seule source de parallélisme disponible car beaucoup d'autres étapes dépendantes du temps sont liées les unes aux autres et nécessitent le calcul d'étapes précédentes pour être réalisées.

L'espace de recherche du problème peut lui aussi être décomposé et une méthodologie particulière employée pour adresser le problème sur chaque composante résultant de la décomposition. L'espace de recherche est partitionné entre différents processeurs et chaque processeur procède à une évaluation séparée du sous espace qui lui est assigné. Malgré le partitionnement, une énumération exhaustive du sous espace n'est généralement pas envisageable et on recourt à une méthode exacte ou à une métaheuristique pour explorer chaque partition. Deux approches peuvent être utilisées pour partitionner l'espace de recherche : la *décomposition de domaine* et la *recherche multiple*.

5.7.1 Évaluation concurrente des voisins

Cette stratégie exploite le potentiel de décomposition intra-algorithme des tâches dans le cadre des boucles d'évaluation des solutions voisines. Elle ne modifie ni la logique de l'algorithme de base, ni l'espace de recherche. Son but est d'accélérer la recherche sans modifier le comportement de la métaheuristique séquentielle.

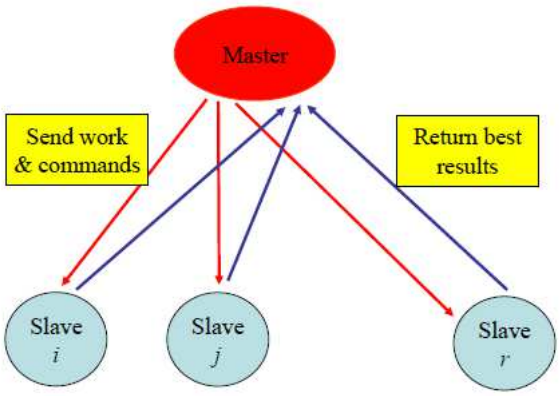
Typiquement, l'exploration est initialisée à partir d'une solution initiale et la recherche se poursuit selon la stratégie de la métaheuristique de base, seules les boucles de calcul internes sont décomposées et les tâches correspondantes sont exécutées simultanément par plusieurs processus.

Cette stratégie est souvent implémentée [58] conformément au modèle parallèle de programmation "*master-slave*" (figure 5.11) : Le programme *master* exécute la métaheuristique conformément au modèle séquentiel en dispatchant en parallèle les tâches de calcul gourmandes sur différents *slaves*. Le programme master reçoit et traite l'information résultant des opérations réalisées par les slaves, sélectionne et implémente les mouvements, actualise la mémoire le cas échéant, et décide de poursuivre la recherche selon différentes stratégies ou de l'arrêter.

Les programmes slaves se contentent de réaliser les évaluations demandées et retournent les résultats au master, qui, lorsque tous les résultats sont disponibles, poursuit la logique normale de la métaheuristique séquentielle. Le contrôle complet de l'exécution repose sur master qui décide d'allouer le travail aux autres processeurs et initie la plupart des communications. Il n'y a pas de communication entre les processus slaves.

[15] Robuste s'entend dans le sens d'offrir un haut niveau de performance uniforme pour une large variété de problèmes et de caractéristiques.

FIGURE 5.11 – Modèle parallèle de programmation Master-Slave d'après [58].



On parallélise la procédure d'évaluation du voisinage pour identifier la solution courante suivante de l'heuristique de recherche locale liée à la métaheuristique. Chaque itération de la boucle génère et évalue un voisin de la solution courante, et peut être exécutée indépendamment des autres itérations. Les calculs de ces itérations peuvent être réalisés sur plusieurs processeurs réservés pour ces tâches. En fait, le master regroupe les voisins dans un nombre approprié de tâches qui sont alors transmises aux slaves. A partir de là, chaque slave peut évaluer chaque voisin appartenant à la partie de son voisinage et retourner le meilleur voisin trouvé. Le master attend que tous les slaves terminent leur calculs, et lorsqu'ils ont tous répondus, sélectionne le meilleur mouvement et poursuit la recherche. GARCIA et al. [101] ont proposé une application de cette stratégie parallèle pour la recherche tabou appliquée à un problème de tournées de véhicules sous contraintes de fenêtres de temps.

Il n'y a pas de taille optimale prédéfinie pour le nombre de tâches parallèles. Le nombre de processus à définir est fortement liées au contexte et dépend entre autres du nombre de processeurs disponibles, des temps de communication inter-processus et de l'architecture matérielle sur laquelle les calculs sont réalisés. Lorsque l'évaluation des voisins est sensiblement la même, il est courant de partitionner les éléments du voisinage en autant de groupes que de processeurs disponibles [58].

Lorsqu'une méthode de descente est utilisé pour retourner le premier meilleur voisin (First Improvement), la procédure de recherche parallèle sera souvent différente de la version séquentielle, et de ce fait, les deux algorithmes se comporteront différemment. De plus, le gain lié à la parallélisation sera limité lorsque beaucoup de bons voisins sont facilement accessibles.

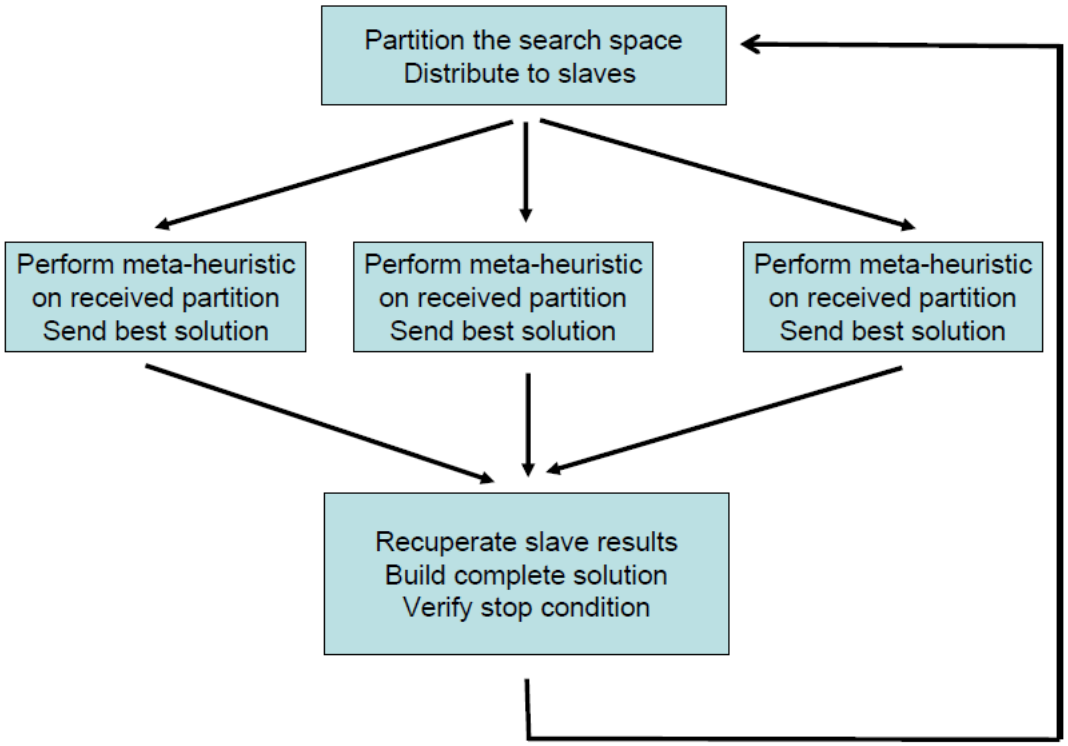
5.7.2 Décomposition de domaine

La décomposition de domaine [227] constitue une stratégie intuitivement simple (en apparence) et attrayante : on divise l'espace de recherche en plus petits ensembles habituellement disjoints et pas nécessairement exhaustifs, on résout les sous-problèmes correspondants en appliquant une métaheuristique séquentielle sur chaque sous ensemble, on collecte les solutions partielles respectives et on reconstruit une solution complète (figure 5.12).

Néanmoins, cette décomposition peut prendre plusieurs formes et un certain nombre de facteurs doivent alors être spécifiés [58] :

1. la partition est-elle stricte ou bien les sous-ensembles peuvent-ils se superposer ?
2. les processus de recherche doivent-ils considérer des solutions complètes ou bien partielles du problème (dans ce dernier cas, une solution complète doit être reconstruire à certains moments) ?
3. les mouvements réalisés sur le sous problème sont-ils restreints au sous espace de recherche correspondant ou bien peuvent-ils intégrer des variables dans les sous espaces voisins créant ainsi une

FIGURE 5.12 – Décomposition de domaine suivant le modèle de programmation Master-Slave d'après [58].



superposition indirecte des sous-ensembles ?

Un partitionnement "*strict*" réduit les processus concurrents de la métaheuristique à leur sous-ensemble respectif et interdit les mouvements impliquant des solutions appartenant à deux sous-ensembles ou plus. Ceci implique qu'une partie de l'espace de recherche est hors de portée et la métaheuristique parallèle devient de fait non optimale. Un "*recouvrement*" permet d'adresser ce problème. Cependant, "*le seul moyen de garantir que toutes les solutions potentielles soient atteignables consiste à faire un chevauchement de la totalité de l'espace de recherche correspondant à une* ***non décomposition****, ce cas étant non pertinent*" [58].

En conséquence, un partitionnement strict ou un chevauchement très limité sont des approches préférées avec une fonctionnalité de re-décomposition généralement incluse pour améliorer la minutie de la recherche et permettre d'examiner toutes les solutions potentielles : la décomposition est modifiée à intervalle de temps régulier et la recherche est relancée en utilisant cette nouvelle décomposition. Cette fonctionnalité permet de définir des décompositions non exhaustives où l'union des sous-ensembles est plus petite que l'espace de recherche complet. Une fonctionnalité de reconstruction d'une solution complète fait presque toujours partie de la procédure.

Le processus master détermine les partitions et transmet les sous ensembles aux slaves, synchronise leur travail et collecte les solutions, reconstruit les solutions (si nécessaire), modifie les partitions, et détermine les conditions d'arrêt (algorithme 10). Chaque slave exécute de façon concurrente et indépendante une métaheuristique séquentielle sur son sous espace de recherche assigné et retourne sa meilleure solution courante après que le master a synchronisé toutes les activités.

Des adaptations sont nécessaires en fonction de la métaheuristique mise en œuvre. Par exemple, Avec l'utilisation d'une recherche tabou, le master doit reconstruire simultanément une liste tabou globale en dehors de la mémoire locale des slaves en lien avec la reconstruction de la solution complète avant de poursuivre la recherche. Le comportement de la recherche et l'effort de calcul réalisé par les métaheuristiques séquentielles et parallèles, tout comme la qualité des solutions respectives différent dans la plupart des cas [268]. Imposer un comportement similaire nécessiterait des efforts importants qui emporteraient tout l'avantage de la parallélisation.

Algorithme 10 : Principe de décomposition de domaine - Processus Master [58]

```
1  Créer la solution initiale s ;
2  Décomposer l'instance du problème en n sous-problèmes : Sp_1, Sp_2, ··· , Sp_n ;
3  while (critère de fin non satisfait) do
4      i ← 1 ;
5      while (i ≤ n) do
6          execute-slave-process(Sp_i, s) ;
7          i ← i + 1 ;
8      Synchronisation des slaves ;
9      i ← 1 ;
10     while (i ≤ n) do
11         s_i ← result-slave-process() ;
12         i ← i + 1 ;
13     s ← construction(s_1, s_2, ··· , s_n) ;
14     Modifier les partitions : Sp_1, Sp_2, ··· , Sp_n ;
15 return s ;
```

5.7.3 Recherche multiple

Dans ce modèle parallèle, des métaheuristiques indépendantes ou coopératives sont utilisées.

La *recherche multiple indépendante* (independent multi-search) consiste à exécuter en parallèle de multiples métaheuristiques indépendantes sur l'espace de recherche entier en initialisant les recherches avec des solutions initiales identiques ou différentes et en sélectionnant à la fin la meilleure solution parmi toutes les recherches. On parle également de *portfolio d'algorithmes*, notion introduite par HUBERMAN et al. [138] qui définit une exécution concurrente de plusieurs algorithmes résolvant un même problème [205]. En plus de la solution initiale, différents réglages de paramètres peuvent être utilisés pour la métaheuristique, comme la taille d'une liste tabou par exemple. De plus, chaque composant de recherche d'une métaheuristique peut être conçu différemment : encodage, opérateur de recherche, fonction objectif, contraintes, critères d'arrêts, etc. [266]. Cette stratégie peut être vue comme une parallélisation directe de l'heuristique multi-start[16]. Cette méthode cherche à accélérer l'exploration de l'espace de recherche en se dirigeant vers une meilleure solution. Aucun avantage n'est attendu des multiples processus qui s'exécutent en parallèle hormis celui d'identifier la meilleure solution globale lorsque tous les processus sont terminés. La recherche multiple indépendante est la stratégie parallèle de recherche la plus simple et offre généralement de bonnes performances, ce qui explique la popularité de cette stratégie pour la parallélisation des métaheuristiques à base de voisinage. On trouve notamment plusieurs exemples de mise en œuvre pour la recherche tabou [228, 265, 31] sur différents problèmes. Le paradigme master-slave est bien adapté à ce modèle : un slave implémente toute la métaheuristique tandis que le master définit les différents paramètres à utiliser par les slaves et détermine la meilleure solution trouvée à partir de toutes les solutions produites par les différents slaves.

Les versions *coopératives* vont plus loin et intègrent des mécanismes de partage pendant la recherche en cours, l'information obtenue permettant de diversifier la recherche. Le comportement global de recherche de la métaheuristique parallèle émerge des interactions locales, ce qui modifie le comportement de la métaheuristique de base. En général, ce partage permet de produire un résultat collectif avec de meilleures solutions qu'une recherche parallèle indépendante.

Pour la mise au point d'une version coopérative, plusieurs questions se posent :

[16] Les méthodes "multi-start" sont des heuristiques itératives composées de deux phases successives : la première phase génère une solution et la seconde améliore cette solution. Chaque itération produit une solution (habituellement un optimum local) et la meilleure solution est retournée à la fin du processus.

- *quand* : quant faut-il échanger de l'information ?
- *où* : compte tenu de la topologie du réseau, quels sont les processus sources et destinations impliqués dans la communication ?
- *quoi* : quelle information pertinente échanger entre les métaheuristiques ?
- *comment* : quel protocole d'échange exploiter ?

5.7.4 Conclusion

La parallélisation permet d'améliorer les performances et la robustesse des métaheuristiques. Il arrive également qu'elle soit mise en œuvre pour améliorer la qualité des solutions ou résoudre des problèmes de grande taille ne pouvant pas être adressés autrement. "*Les métaheuristiques devenant de plus en plus efficaces, elles sont de plus en plus souvent appliquées à la résolution de problèmes de très grande taille. On retrouve là les deux principales motivations classiques de l'utilisation du parallélisme pouvant être résumées par* ***plus vite*** *et* ***plus grand***" [187].

Trois formules principales plus ou moins invasives sont identifiées dans la littérature pour paralléliser les métaheuristiques. On peut procéder à une évaluation concurrente du voisinage, à une décomposition du domaine ou à une recherche multiple (indépendante ou coopérative).

Bien que les principes de parallélisation puissent paraître simples, la mise au point peut révéler des aspects très complexes. Elle doit notamment tenir compte de l'infrastructure matérielle, des réseaux disponibles, des systèmes d'exploitation, etc. Les protocoles de communication, les informations échangées et les modes de synchronisation entre processus doivent être appropriés car ils peuvent inhiber tout le bénéfice des traitements parallèles. D'autre part, la structure même d'une métaheuristique se prête plus ou moins bien à la décomposition en tâches indépendantes. Et il peut être très difficile de reproduire dans la version parallèle d'une métaheuristique le comportement exacte de sa version séquentielle. Par exemple, du fait de la structure séquentielle intrinsèque au recuit simulé, la parallélisation en l'état de cette méthode peut sembler paradoxale. Néanmoins, des modes de fonctionnement parallèles plus ou moins complexes de cette métaheuristique ont tout de même été proposés [245].

La plupart du temps, la parallélisation des métaheuristiques permet d'obtenir facilement de très bonnes accélérations [187]. Par exemple, pour rendre la méthode tabou parallèle, il suffit de répartir l'examen du voisinage de la solution courante sur différents processeurs. La marche dans l'espace de recherche est généralement très peu modifiée par rapport à la version séquentielle mais elle est effectuée plus rapidement grâce aux calculs répartis sur plusieurs processeurs.

5.8 Environnements logiciel et cadre générique

La distinction entre "aspect problème-indépendant" intrinsèque aux métaheuristiques et "aspect problème-dépendant" lié aux heuristiques de recherche révèle un atout majeur : les deux aspects peuvent être réalisés séparément l'un de l'autre. Il devient possible d'implémenter les stratégies génériques de façon abstraite une fois pour toutes dans le cadre d'un framework (éventuellement orienté objet). Face à un problème concret, seules les heuristiques spécifiques devront être implémentées pour être connectées au framework.

Plusieurs librairies et frameworks ont été développés pour simplifier la mise en œuvre de métaheuristiques et la réutilisation du code. Le lecteur intéressé par une liste complète des librairies et frameworks disponibles pourra consulter [266, 146] pour obtenir leurs principales caractéristiques.

5.8.1 Les origines

Les premières études qui ont pour ambition d'unifier les différents types d'algorithmes de RL et d'améliorer leur compréhension remontent à la fin des années 90. Dans [290], YAGIURA et IBARAKI proposent d'adopter un cadre générique pour comparer les comportements des divers algorithmes et leurs raffinements. Plus

générale et originale, l'approche de TAILLARD et al. [264] utilise le concept *Adaptive Memory Programming* pour fédérer recherche tabou, algorithmes génétiques (AG), recuit simulé et même colonie de fourmis. Par exemple, la population d'un AG est vue comme la mémoire de la recherche du passé, au même titre qu'une liste tabou. Ce formalisme permet également de décrire et mieux comprendre le fonctionnement hybride de ces métaheuristiques.

Les outils destinés à abstraire les mécanismes élémentaires de la résolution par RL sont apparus avec *Localizer* [191], un langage de modélisation de haut niveau dédié à la spécification d'algorithmes de RL et de ses structures de données qui permet d'exprimer de manière souple et concise la résolution d'un problème. Le système maintient automatiquement et efficacement les structures de données lors de la recherche. Les algorithmes de recherche locale peuvent être exprimés dans une notation proche de leurs descriptions informelles présentes dans les papiers scientifiques.

Ensuite, des librairies de classes génériques ou des frameworks sont apparus. Parmi eux, on peut citer par exemple : *Comet*, *EasyLocal++*, *HotFrame*, *LocalSolver* ou encore *ParadisEO*.

5.8.2 Bibliothèques et frameworks

Comet *Comet* [272, 183], qui fût la plateforme de recherche du laboratoire d'optimisation de la BROWN UNIVERSITY, est un outil qui a été primé pour résoudre des problèmes complexes d'optimisation combinatoire dans des domaines tels que l'allocation de ressources et l'ordonnancement. Il peut être vu comme une hybridation d'algorithmes de recherche locale et de programmation par contraintes. En effet, une des principales innovations de Comet repose sur le concept de recherche locale à base de contraintes (Constraint-Based Local Search) : ce paradigme permet de construire des algorithmes de recherche locale en séparant la modélisation de la recherche, et de promouvoir ainsi la réutilisation. Comet est basé sur un langage spécifique orienté objet très riche qui supporte à la fois la modélisation et l'abstraction des recherches dans l'esprit de la programmation par contraintes. Sa force peut être exprimée par la formule : "*Recherche Locale = modèle + recherche*" signifiant que les algorithmes de recherche locale peuvent être spécifiés en termes de modélisation et de composants de recherche. Les deux composantes "contraintes" et "recherche" sont présentes et dissociées. Comet propose un langage de contraintes et des constructions pour programmer des stratégies de branchement, des heuristiques d'ordonnancement (de listes), et aussi des parcours d'arbres. Le langage offre des facilités pour l'ingénierie et permet par exemple différentes entrées / sorties pour se connecter à des bases de données ou lire des fichiers XML. Comet permet aussi de construire des interfaces utilisateur pour observer graphiquement l'évolution de l'exécution des modèles compilés. Enfin, le langage prend le parti de programmer la recherche. Il offre des structures de contrôle comme les conditionnelles et les boucles « tant que » et demande à déclarer, transformer ou manipuler les structures de données de telle sorte qu'elles soient exploitables par des heuristiques. Malheureusement, cette plateforme ne semble plus maintenue. De plus, son langage est dédié et Comet n'est pas disponible en open-source.

EasyLocal++ *EasyLocal++* [74] est une librairie open-source orientée objet pouvant être utilisée pour développer des algorithmes de recherche locale en *C++*. Elle offre une liberté totale à l'utilisateur pour définir ses structures de données et les domaines des variables, mais fournit un cadre rigide pour contrôler le flot d'exécution des principales métaheuristiques qu'elle propose en standard. EasyLocal++ fait un usage équilibré de la programmation orientée objet utilisée pour la conception du framework. Les classes de données sont fournies par des templates et les structures algorithmiques sont implémentées par des fonctions virtuelles, offrant ainsi un maximum d'efficacité, de contrôle et de souplesse pour une implémentation incrémentale en niveaux hiérarchiques. La modularité est l'une de ses principales caractéristiques : dès que les structures de données et les opérations de base ont été définies et connectées, le système fournit en libre service toutes les techniques standards de RL et une grande variété de combinaisons constituée à partir de ces techniques de base. Le système permet également de construire et de tester de nouvelles combinaisons de fonctionnalités (e.g. structures de voisinage, stratégies pour générer un état initial, mécanismes d'interdiction de mouvement, ...). Ceci permet à l'utilisateur de concevoir et d'implémenter facilement

de nouvelles heuristiques de façon très structurée pour adresser un problème donné. Ce framework peut être personnalisé par un expert pour développer de nouvelles métaheuristiques, et son architecture facilite grandement la réutilisation du code existant. On retrouve dans la librairie des classes pour supporter les tests algorithmiques. Certaines classes permettent également de collecter des statistiques d'exécution, les informations de sortie pouvant être générées dans différents formats.

HotFrame *HotFrame* [93] (Heuristic OpTimization FRAMEwork) est un framework implémenté en C++ qui propose des algorithmes de recherche locale et des algorithmes à base de population de solutions. HotFrame repose sur une architecture de collaboration entre les composants standards et les heuristiques spécifiques aux problèmes. Tous les concepts classiques spécifiques aux problèmes sont représentés par des objets ou des classes : problème, solution, voisin, voisinage et mouvement. C'est la même chose pour les concepts standards liés aux métaheuristiques, à savoir les différentes méthodes ou stratégies (intensification, diversification). HotFrame utilise la généricité comme mécanisme de base pour rendre ces objets adaptables : le comportement commun des métaheuristiques est factorisé et regroupé dans des classes génériques. Les templates des classes représentant les métaheuristiques sont paramétrables selon différents aspects comme les espaces solutions et les structures de voisinage. Tous les composants sont implémentés de manière cohérente, ce qui facilite l'intégration d'heuristiques liées aux problèmes ou la conception de nouvelles méthodes. L'utilisateur peut dériver des classes existantes ou implémenter de nouvelles classes à partir de zéro suivant une interface prédéfinie. HotFrame propose également des structures de voisinage standards, la représentation de solutions et de mouvements ainsi que des opérateurs de recombinaison. En proposant ce framework, les concepteurs ont recherché à la fois la performance et la facilité de réutilisation du code. La dernière version disponible[17]remonte à juillet 2003.

LocalSolver *LocalSolver* [103] est un solveur de programmation mathématique qui intègre la technologie de recherche locale. LocalSolver est commercialisé par la société *Innovation24*[18] sous la forme d'une "boite noire". L'utilisateur modélise son problème dans un formalisme simple. Le solveur se charge d'exploiter efficacement ce formalisme pour résoudre le problème sans qu'il soit nécessaire de programmer un mouvement ou une recherche. Le langage dédié proposé (LSP) se veut adapté pour modéliser simplement une grande variété de problèmes. Des Api *C++*, *C#* et *Java* sont également disponibles.

ParadisEO *ParadisEO* [38] est une plateforme logicielle orientée objet dédiée à la conception et à la réutilisation de métaheuristiques. Il s'agit d'un framework open-source qui repose sur la notion de template *C++*. Il est portable, très complet, et basé sur la librairie *Evolving Objects*[19] (voir la figure 5.13). ParadisEO supporte les métaheuristiques de recherche locale ou à base de population de solutions, l'optimisation mono-objectif et multi-objectif ainsi que le parallélisme.

5.9 Conclusion

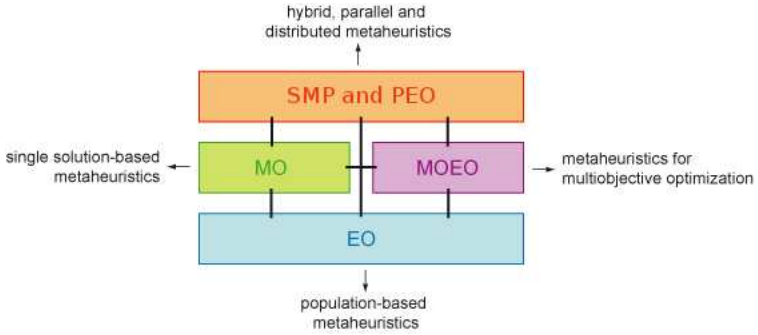
Nous avons abordé ce chapitre central en commençant par définir les caractéristiques principales des problèmes combinatoires avant de recenser les méthodes disponibles pour résoudre ces problèmes souvent difficiles. Nous avons vu qu'il existait deux grandes classes de méthodes adaptées à leur résolution : les

[17]Libre de droits pour une utilisation non commerciale, Institute of Information Systems de Hambourg : http://www1.uni-hamburg.de/IWI/hotframe/hotframe.html.

[18]Filiale du Groupe BOUYGUES issue de son département de R&D (e-lab), il s'agit d'une société de services, éditrice de logiciels dans le domaine de la recherche opérationnelle et de l'aide à la décision (http://www.innovation24.fr/).

[19]EO : framework pour l'implémentation de métaheuristiques à base de population. Propose des algorithmes évolutionnaires et des algorithmes à essaim de particules (http://eodev.sourceforge.net/).

FIGURE 5.13 – Modules de la plateforme ParadisEO d'après [146].



méthodes complètes d'une part qui obtiennent les solutions optimales et garantissent les résultats et les méthodes incomplètes d'autre part qui trouvent rapidement de bonnes solutions sans garantir l'optimalité.

Face à ces nombreuses méthodes de résolution disponibles, le premier problème pratique qui se pose à un utilisateur confronté à une application concrète est d'effectuer un choix parmi les techniques les plus adaptées. Or ce choix est d'autant plus difficile qu'il ne semble pas exister de comparaison générale et fiable entre les différentes techniques, ni même entre les différentes métaheuristiques.

Néanmoins, notre étude permet de faire un premier constat : il n'existe pas une méthode qui soit la meilleure pour résoudre tous les problèmes. D'autre part, il est essentiel de prendre en compte, en plus du problème, les contraintes qui entourent sa résolution, notamment le critère de temps de réponse qui sera déterminant par la suite.

De plus, cet état de l'art permet de dégager des règles générales qui semblent s'imposer. Lorsqu'un problème donné peut être résolu avec une méthode exacte tout en répondant aux besoins contextuels, il faut privilégier ces techniques qui garantissent les résultats obtenus.

Toutefois, pour de nombreux problèmes, il arrive que les méthodes complètes ne soient pas en mesure de délivrer un résultat dans le temps imparti. Dans ce cas, si la situation le permet, il est intéressant d'exploiter ces méthodes complètes au moins pour cerner la difficulté du problème et identifier des solutions optimales avec les coûts correspondants (valeur de la fonction objectif). Ensuite, la mise en œuvre d'une métaheuristique permettra d'obtenir de bons résultats, même pour des instances de taille importante. Néanmoins, si le problème est de très grande taille et si le temps de résolution devient trop important, les mécanismes distribués constituent une alternative efficace pour repousser encore ces limites.

6

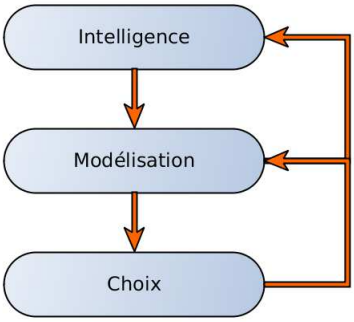
Systèmes interactifs d'aide à la décision

Nous abordons la conception interactive sous l'angle d'une plateforme d'aide à la décision reposant sur des composants informatiques en charge de fournir aux décideurs le maximum d'éléments dans les meilleurs délais pour les aider à résoudre des problèmes complexes auxquels ils sont confrontés. La complexité des systèmes informatiques mis en place pour explorer, traiter et présenter des informations capables d'aider l'utilisateur dans ses décisions est une question de recherche difficile qui reste ouverte. Nous nous intéressons plus particulièrement au temps de traitement ainsi qu'à la qualité des solutions proposées par les applications de résolution sous-jacentes qui doivent répondre très rapidement aux sollicitations des utilisateurs pour garantir la fluidité de l'interaction homme/machine. Concevoir un système d'aide à la décision, c'est concevoir un système temps réel [258] capable de fournir des résultats dans un délai contrôlé. Or, obtenir des résultats dans un délai très court fixé à l'avance reste très difficile pour des problèmes à forte combinatoire. Néanmoins, dans le contexte des systèmes d'aide à la décision, une bonne solution obtenue dans les temps est souvent préférable à un résultat optimal obtenu hors délais. L'approche de résolution *anytime* que nous abordons dans ce chapitre permet de fournir des résultats de meilleure qualité à chaque pas de temps faisant que plus on accorde de temps à la procédure, meilleur est le résultat [83]. Cette approche permet de prendre en compte nos exigences (en qualité et en temps liées aux SIAD) et apparaît donc pertinente pour répondre à notre problématique [237, 291].

6.1 Aide à la décision

Les décisions individuelles ou collectives prises au sein d'une organisation par les décideurs participent d'une manière ou d'une autre à son devenir. Elles sont le plus souvent prises sur la base d'intuitions et d'expériences du passé [4]. Comme indiqué par H.A. SIMON [249], ces stratégies ne peuvent s'appliquer qu'à des problèmes familiers. En effet, la prise de décision devient nettement plus difficile face à des situations nouvelles et la difficulté augmente encore avec l'évolution rapide d'un environnement de travail de plus en plus complexe. Pour HOLTZMAN [134], l'un des principaux problèmes pour la prise de décision consiste à déterminer les informations pertinentes. Pour prendre les bonnes décisions face à des circonstances nouvelles et compliquées, l'utilisation des Systèmes Interactifs d'Aide à la Décision devient essentielle afin d'évaluer la situation, les différentes alternatives et leurs impacts.

FIGURE 6.1 – Modèle IDC (Intelligence, Design, Choice) proposé par H.A. SIMON [249].



6.1.1 Décision

Pour nombre de chercheurs, la décision peut être perçue comme un choix entre plusieurs alternatives [242]. P. LÉVINE et J.C. POMEROL [171] la définissent comme une action prise pour faire face à une difficulté ou répondre à une modification de l'environnement pour résoudre un problème qui se pose à l'individu ou à l'organisation. B. ROY et D. BOUYSSOU [236] pensent que la décision est le fait d'un individu isolé (le décideur) qui exerce librement un choix entre plusieurs possibilités d'actions à un moment donné. Pour H. MINTZBERG [193], une décision individuelle ou collective peut être définie comme l'engagement dans une action correspondant à une intention explicite d'agir.

6.1.2 Processus de décision

Herbert Alexander SIMON, qui a reçu le prix Turing en 1975 puis le prix Nobel d'économie en 1978, a proposé en 1960 un schéma général du processus de décision [1] (voir la figure 6.1) en trois phases qui reste encore le modèle de référence [249] :

1. **phase d'intelligence (investigation)** : il s'agit de recenser l'ensemble des informations utiles et prioritaires dont le décideur aura besoin lors de la prise de décision (objectifs ou priorités, définition du problème à résoudre).
2. **phase de modélisation (conception)** : concerne la génération et le développement des différentes alternatives (ou scénarios). Cette étape peut nécessiter une recherche d'information complémentaire.
3. **phase de choix (sélection)** : le décideur évalue les différents scénarios réalisés pendant la phase de conception pour en retenir un seul. Il s'agit de déterminer les critères d'évaluation des différentes solutions possibles et de mesurer les conséquences de chaque alternative.

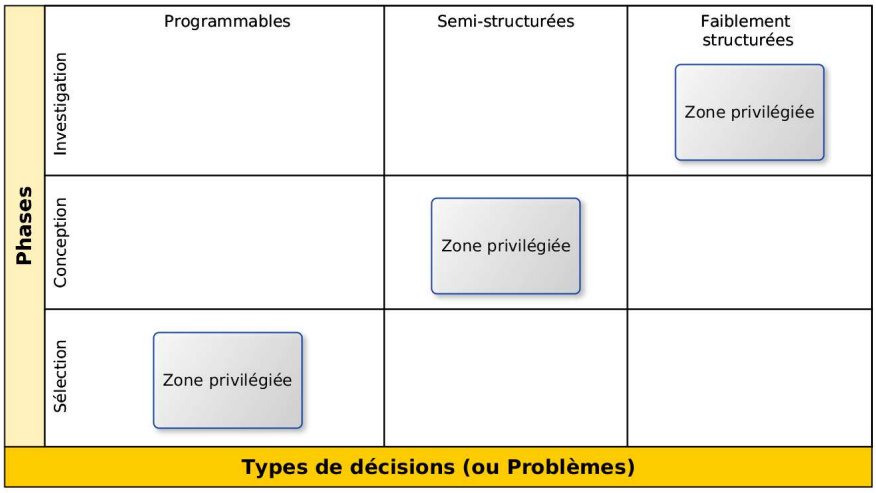
Bien que ce modèle ne reflète que partiellement la réalité, il permet d'avoir une vision globale des différentes tâches à accomplir pour aboutir à une décision. Les différentes phases se succèdent sans logique chronologique préétablie et la présence de boucles pendant le processus de décision dépend du niveau de structuration du problème (contraintes). Dans la figure 6.2, LE MOIGNE [167] met en correspondance l'importance des phases du modèle IDC au regard de la structuration des décisions (problèmes).

6.1.3 Aide à la décision

Pour B. ROY [235], aider à décider consiste tout d'abord à aider à clarifier la formulation, la transformation et l'argumentation des préférences, la notion de critère étant au cœur de la démarche. L'activité de déduction et de modélisation qui s'exerce en vue d'éclairer le comportement d'un individu dans le processus de

[1] Dans la suite de cette section, nous n'aborderons pas l'aspect mise en œuvre d'une décision.

FIGURE 6.2 – Correspondance entre l'importante des phases du modèle IDC et la structuration des décisions proposée par LE MOIGNE [167].



décision constitue l'essence même de l'aide à la décision [236]. L'objectif n'est donc pas de résoudre le problème, mais d'aider le décideur à construire une représentation pertinente de la situation, sachant qu'il n'existe pas toujours de solution optimale pouvant s'imposer compte tenu de la multiplicité des points de vue et des intérêts défendus ou du caractère multidimensionnel des contextes en situation de décision. Dans ce cadre, la notion de meilleure décision (solution optimale) peut parfois être vide de sens [240].

L'aide à la décision peut néanmoins reposer sur des techniques et des méthodologies issues du domaine des mathématiques appliquées telles que l'optimisation, les statistiques ou la théorie de la décision. Les décisions peuvent être classées en plusieurs groupes [193]. Certaines sont faciles à prendre et se font par ajustement mutuel (via le dialogue) tandis que d'autres obéissent à un déroulement fixe. Ces dernières intègrent parfois un aspect purement combinatoire pouvant être automatisé et confié à une machine. Cependant, la majorité des décisions combinent les deux aspects à la fois : le dialogue et la combinatoire. Pour ce type de problème, la machine peut devenir un acteur à part entière dans la prise de décision, mais elle ne doit en aucun cas être perçue comme une contrainte supplémentaire qui imposerait ses propres décisions sans laisser de marge suffisante aux vrais acteurs que sont les humains. L'aide à la décision réside sur l'équilibre entre le jugement humain et le traitement informatique.

Lorsque l'évaluation globale d'un objectif est complexe, il est nécessaire de décomposer l'objectif à atteindre en structurant l'ensemble des critères d'évaluation [219]. Construire un modèle prenant explicitement appui sur plusieurs critères traduit et formalise un mode de raisonnement intuitif et naturel face à un problème de décision qui consiste à analyser séparément chaque conséquence [234].

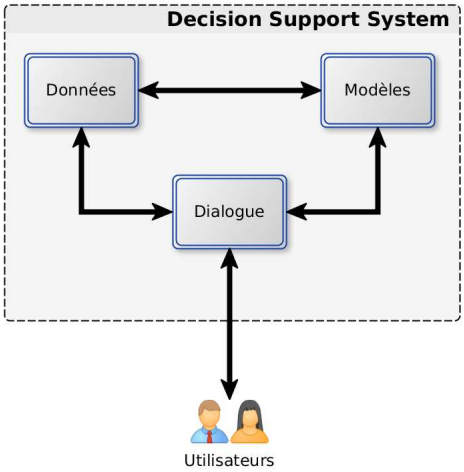
6.1.4 Systèmes Interactifs d'Aide à la décision

Les premiers outils d'aide au traitement de l'information et à la prise de décision sont apparus il y a une quantaine d'années [170]. En France, ils sont connus sous le nom de « Système d'Aide à la Décision » (SAD) alors que dans les pays anglo-saxons on parle de « Decision Support System » (DSS) [171]. Plus qu'un simple système de traitement de l'information permettant d'extraire et de donner au décideur l'information nécessaire au processus de décision [171], un SAD peut être considéré comme un résolveur de problèmes (*problem solver*) [204].

Un SAD se compose de trois modules (figure 6.3) :

- un module de dialogue : permet au décideur d'accéder aux données et modèles et lui fournit en retour le résultat de ses manipulations pour exercer son contrôle et sa recherche heuristique dans les

FIGURE 6.3 – Structure d'un SAD proposée par SPRAGUE [255].



meilleures conditions ;
- un module de données : stocke les données permanentes (statistiques, données qui décrivent la situation courante ou passée, etc.) ou temporaires (données intermédiaires) ;
- un module contenant les procédures de calcul ou modèles : procédures de calculs standards (indices, indicateurs, etc.) et procédures de représentation des données.

Le concept d'aide à la décision "interactive" fait référence à l'équilibre entre le jugement humain et le traitement des informations par l'ordinateur. On parle alors de « Système Interactif d'Aide à la Décision » (SIAD). Lorsque le SIAD intègre en plus une dimension spatiale (représentation de cartes géographiques, plans, etc.), on est face à un « Système Spatial Interactif d'Aide à la Décision » (Spatial Decision Support System en anglais). Les SIAD recouvrent un large spectre d'applications (potentiellement chaque activité humaine nécessitant un processus de décision élaboré) et ils s'appuient sur plusieurs domaines de recherche de l'informatique comme par exemple les systèmes de gestion de bases de données (SGBD), la recherche opérationnelle ou l'intelligence artificielle.

Dès leur origine, on pouvait distinguer deux catégories de SIAD [50] :

- ceux qui font une large place aux algorithmes d'optimisation et aux calculs numériques ;
- ceux qui se consacrent exclusivement à la gestion de l'information.

P.G.W. KEEN et M.S. SCOTT-MORTON [149] présentent les SIAD comme des systèmes adaptés à la résolution des problèmes de décision peu ou mal structurés où l'homme prend l'avantage sur la machine, contrairement au traitement des problèmes structurés où la machine peut prédominer. Voici la définition qu'ils en donnent : « *Les systèmes d'aide à la décision permettent de coupler les ressources intellectuelles des individus avec les capacités des ordinateurs pour améliorer la qualité des décisions. C'est un système d'aide informatique aux décideurs qui traitent des problèmes semi-structurés* ». Pour R.H. SPRAGUE et E.D. CARLSON [254], « *Les SIAD sont des systèmes informatisés et interactifs qui aident les décideurs en utilisant des données et des modèles pour résoudre des problèmes mal structurés* ».

Il existe un large consensus pour dire que les SIAD doivent posséder des facilités interactives afin d'interroger l'utilisateur. La notion d'interactivité est une caractéristique commune à tous ces systèmes qui renvoie au rôle actif indispensable de l'utilisateur dans leur fonctionnement au travers d'une interface homme/machine bien intégrée [171]. L'information nécessaire à l'utilisateur doit être disponible sous sa forme la plus adéquate pour qu'il puisse prendre des décisions. En permettant d'éliminer les données non pertinentes et en présentant des faits déduits à partir des systèmes d'information, l'intelligence artificielle a joué très tôt un rôle important dans ce dispositif et son lien avec les SIAD n'a fait que se renforcer au cours des années [221, 56]. Le décideur contrôle le processus de décision et le SIAD l'assiste en effectuant les calculs standards et répétitifs sur les données [151].

Lorsque l'utilisateur est l'élément prépondérant du couple homme-machine, tout ou partie du processus de décision lui revient. Toutes les solutions ne sont pas atteintes et seul un sous-espace de l'espace de recherche est exploré. On parle de *recherche heuristique* [2] menée par le décideur avec un système qui jalonne le processus de recherche à l'aide d'indicateurs et d'informations. Le décideur stoppe l'exploration heuristique des actions possibles lorsque tout lui indique que la solution construite rencontre ses buts de façon satisfaisante [221, 171]. La coopération entre le décideur et le système informatique ne peut alors être fructueuse que dans le cadre d'un système interactif.

Enfin, pour qu'un SIAD soit accepté par ses utilisateurs, il faut non seulement qu'il puisse gérer toutes les informations utiles à la décision, mais il faut également que sa logique décisionnelle puisse être appréhendée par ceux qui l'utilisent : l'explication permet en effet de juger de la pertinence de ses déductions ou de ses inférences. Comme nous l'avons déjà évoqué, l'objectif n'est pas de rechercher une solution optimale, mais plutôt d'orienter le décideur vers des points qu'il ne serait pas toujours en mesure d'observer sans une assistance appropriée.

6.1.5 Classification des SIAD au niveau utilisateur

P. HÄTTENSCHWILER [98] énumère trois catégories de SIAD pouvant être observées du point de vue des utilisateurs :

1. **passif** : aide le processus décisionnel mais n'est pas en mesure d'apporter de solutions ou de suggestions de décisions explicites ;
2. **actif** : peut apporter des solutions ou suggestions de décisions explicites ;
3. **coopératif** : permet au décideur de modifier, de compléter ou d'affiner les suggestions de décisions fournies par le système avant de les renvoyer au système pour validation. De son coté, le système améliore à son tour, complète et affine les suggestions du décideur avant de les lui renvoyer pour validation. Cette boucle de rétroaction se poursuit jusqu'à ce qu'une solution consolidée soit générée.

6.2 Résolution temps réel

Les applications temps réel sont présentes dans un nombre croissant de domaines comme l'industrie, la médecine, les communications, l'aéronautique, la robotique, les transports ou l'armement [243]. Plus récemment, ces systèmes ont fait leur apparition dans bien d'autres secteurs, notamment le multimédia, les consoles de jeux ou le suivi des malades [83]. La notion d'application temps réel est très souvent utilisée sans qu'elle ne soit toujours bien définie. On peut néanmoins qualifier une application temps réel comme une application dont l'exécution est soumise à des contraintes temporelles. Cindy A. O'REILLY et Andrew S. CROMARTY [208] identifient un système temps réel comme un système devant donner une réponse en un temps strict au regard de l'algorithme mis en œuvre. Pour Thomas J. LAFFEY et al. [159], c'est la possibilité pour un système de garantir une réponse après qu'un certain temps se soit écoulé. On peut également percevoir un système temps réel comme un système dont le temps de réponse est rapide [172]. Or selon MUSLINER et al. [201], une application temps réel ne correspond pas à un système "rapide" mais plutôt à un système "suffisamment rapide" pour répondre en un temps donné, ce temps étant lié au contexte de résolution. Dans [243], lorsqu'on étudie un système de décision temps réel (Real-Time Decision System en anglais ou RTDS), la première chose qui vient à l'esprit est probablement la notion de réaction rapide à un événement externe.

Compte tenu de leur grande diversité, une classification courante de ces systèmes consiste à évaluer les conséquences provoquées par le non respect de l'échéance :

- échéance **stricte critique** : le manquement de l'échéance peut avoir des conséquences catastrophiques (pertes humaines, matériel, environnement, ...) ;

[2] Une méthode heuristique est une méthode dans laquelle, à chaque pas, le raisonneur fait des choix [56].

- échéance **stricte non critique** : lorsque l'échéance n'est pas respectée, la tâche devient caduque ;
- échéance **non stricte** : le dépassement de l'échéance est sans conséquence et la tâche peut se poursuivre, seule la qualité du service sera diminuée.

Dans cette thèse, nous abordons la notion de résolution temps réel dans le cadre d'un SIAD en lien avec des problèmes d'optimisation combinatoire dont l'échéance pourrait être considérée comme *stricte non critique* voire *non stricte*. Pour un tel système interactif mettant en relation un utilisateur avec une application chargée de l'accompagner dans ses décisions, l'application en question doit pouvoir réagir aux événements de l'utilisateur en une fraction de seconde. Or, si un événement de cette application concerne la résolution d'un problème combinatoire sur des instances réelles de grande taille, il parait difficile à première vue d'atteindre l'objectif de temps fixé. Néanmoins, en admettant que notre application puisse utiliser des résultats dont la qualité n'est pas optimale, nous pouvons explorer une voie prometteuse relative aux algorithmes *anytime* ("à tout moment") apparus en intelligence artificielle à la fin des années 80 [67, 136].

6.2.1 Algorithme *anytime*

T. DEAN et M. BODDY [67, 32] définissent un algorithme *anytime* comme un algorithme itératif qui garantit de produire une réponse à toute étape de calcul, où la réponse est supposée s'améliorer à chaque itération. Du point de vue de l'implémentation, un algorithme *anytime* doit posséder les caractéristiques suivantes [83] :

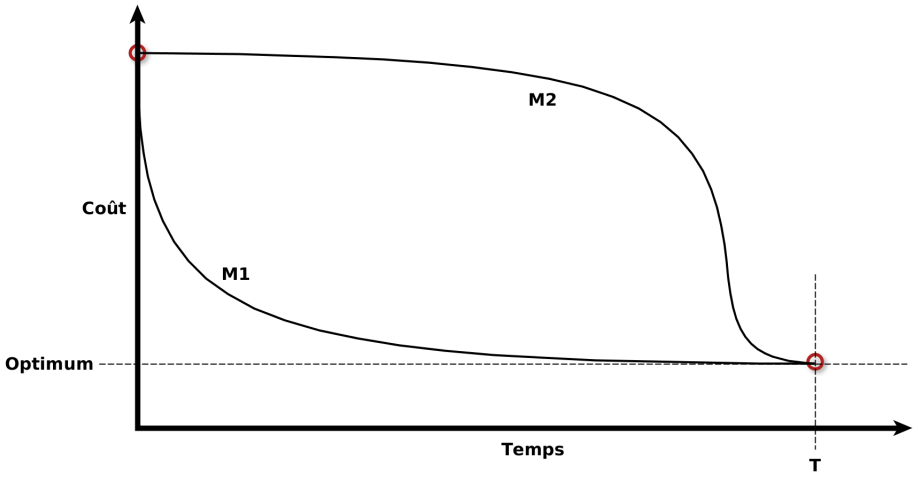
- la qualité du résultat est fonction du temps d'exécution alloué et de la qualité des ressources fournies en entrée ;
- la qualité du résultat augmente toujours ou reste stable en fonction de l'évolution du temps de calcul (monotonicité) ;
- on dispose d'informations statistiques sur la qualité des résultats en fonction du temps alloué (prévisibilité) ;
- il peut être interrompu à tout moment (interruptibilité) ;
- il peut poursuivre son exécution au delà du temps alloué si on change le temps prévu en cours d'exécution (continuité) ;
- il doit pouvoir être suspendu et relancé au même point avec un temps de réponse négligeable.

On distingue deux types d'algorithme *anytime* : celui où le temps est connu avant de lancer la résolution et celui où le temps n'est pas défini à l'avance. Si le temps de résolution est connu au lancement, on parle de *contexte contractuel* et le processus de résolution doit absolument fournir une solution au terme de l'échéance fixée. Dans le cas où le temps de résolution n'est pas connu à l'avance, on parle de *contexte interruptible*. Dans une telle situation, l'algorithme de résolution doit être en mesure de fournir à tout moment une solution de la meilleure qualité possible. Être dans la capacité de fournir à chaque instant un résultat quel que soit le temps disponible représente une exigence très forte. Néanmoins, dans la pratique, le temps de résolution attribué au contexte doit toujours être supérieur à un temps minimum permettant au processus d'initialiser la recherche et de générer une première solution [243]. Un algorithme contractuel (ou algorithme *anytime* contractuel) est plus facile à mettre au point qu'un algorithme interruptible (ou algorithme *anytime* interruptible). Cependant, un algorithme interruptible peut être construit à partir d'un algorithme contractuel au prix tout de même d'un temps de calcul supplémentaire pour atteindre une solution de même qualité.

6.2.1.1 Qualité des solutions

Dès qu'une limite de temps est imposée, il n'est plus certain que la résolution puisse aller jusqu'à son terme. Dans ce cas, le résultat peut être partiel (sous-optimal) ou même incorrecte si la solution produite n'est pas admissible. Il peut donc être utile ou même indispensable dans certaines circonstances de connaître

FIGURE 6.4 – Comparaison des *profils de qualité* de deux méthodes de résolution **M1** et **M2** sur une même instance de problème. Bien que les deux méthodes atteignent l'optimum pour un temps **T** de résolution identique, la méthode **M1** apparaît plus efficace que **M2** dans un contexte interruptible. D'après la thèse de Lionel LOBJOIS [172].



précisément ou d'évaluer la qualité d'une solution afin de prendre une décision. Par exemple, dans un contexte interruptible, l'utilisateur pourrait décider de ne pas poursuivre une résolution si la qualité de la solution courante est satisfaisante. Connaître le coût d'une solution n'est pas toujours suffisant pour évaluer la qualité de celle-ci. On doit en effet pouvoir évaluer sa distance à l'optimum. Cependant, la distance d'une solution à l'optimum est aussi difficile à trouver que l'optimum lui même. Compte tenu de cette difficulté, on se limite très souvent dans la pratique à rechercher la meilleure solution possible pour un contexte de résolution donné. Cet objectif consiste alors à trouver une solution du meilleur coût possible sans pouvoir quantifier sa qualité [172].

6.2.1.2 Profil de qualité

Le *profil de qualité* (PDQ) correspond à la courbe d'évolution du coût de la meilleure solution produite par rapport à la progression du temps de résolution [172]. Le PDQ d'une méthode de résolution fournit beaucoup plus d'indications qu'un simple temps de résolution global et il est particulièrement utile pour évaluer une méthode de résolution dans un contexte interruptible. A titre d'exemple, imaginons deux méthodes M1 et M2 (voir la figure 6.4) capables de fournir toutes les deux une solution optimale sur la même instance de problème pour la même échéance de temps. Dans un contexte interruptible, il semble évident que la méthode M1 est plus efficace que la méthode M2 car elle produit à tout moment des solutions de meilleure qualité. Cependant, il peut être difficile de comparer deux PDQ différents lorsque chaque méthode propose tour à tour des solutions de meilleure qualité. Dans ce cas, une issue consiste à fixer soit un seuil de qualité soit un seuil de temps pour permettre la comparaison. On utilisera le plus souvent un *PDQ moyen* compte tenu du caractère stochastique de certaines méthodes pouvant produire pour une même instance de problème des PDQ très différents. Le *PDQ moyen* correspond à la courbe de qualité moyenne de la meilleure solution obtenue en fonction du temps de recherche imparti. Dans le cadre des problèmes d'optimisation combinatoire et en l'absence d'information sur la qualité des solutions produites, il est courant d'utiliser le *PDQ moyen* pour classer les méthodes de résolution.

6.2.2 Méthodes de résolution adaptées au contexte de décision temps réel

Pour R. SÉGUIN et al. [243], lorsque les temps de résolution imposés sont très courts, les méthodes approchées (métaheuristiques, voir la section 5.2.2) constituent des classes d'algorithmes particulièrement appropriées. Plus précisément, les algorithmes génériques issus de la recherche opérationnelle tels que le *recuit simulé* (section 5.6.7), la *recherche tabou* (section 5.6.8), la méthode *GRASP* (section 5.6.10.1) ou encore les *algorithmes génétiques* possèdent tous certaines caractéristiques requises pour la résolution de problèmes de décision temps réel. Chacune des quatre méthodes expose notamment le comportement des algorithmes *anytime* avec une qualité de la solution produite qui s'améliore avec le temps. Même si au sens strict du terme ils ne peuvent pas être considérés tels quels comme des algorithmes *anytime* car incapables de produire une solution initiale si le temps imposé est insuffisant, ils peuvent néanmoins être facilement adaptés pour répondre au cas par cas à l'ensemble de caractéristiques requises.

D'un autre côté, les méthodes complètes (section 5.2.1) semblent peu adaptées à cet environnement compte tenu de leur côté systématique et exhaustif [172]. Les méthodes de résolution basées sur une approche de construction (section 5.5) ne sont pas adaptées non plus au contexte *anytime* car elles construisent pas à pas une solution en partant d'une solution partielle initialement vide qu'elles cherchent à étendre à chaque étape jusqu'à obtenir une solution complète [127], ce qui écarte de nombreuses méthodes comme les méthodes gloutonnes (section 5.5.1), les méthodes avec retour arrière (section 5.4.1.4) ou encore celles basées sur des algorithmes de séparation et d'évaluation (section 5.2.1.2) progressive.

6.2.3 Contribution du parallélisme

Dans un environnement temps réel, et bien que cela ne soit pas une démarche naturelle, il est souvent nécessaire de mettre en œuvre des techniques de calcul parallèle pour améliorer la vitesse des traitements [243]. On constate en effet une activité de recherche importante autour de l'implémentation d'heuristiques de recherche parallèles performantes (voir 5.7). Mais une telle démarche est souvent ardue. Par exemple, l'algorithme de *recherche tabou* est à la base un algorithme séquentiel. On peut alors exploiter le parallélisme pour permettre à différents processeurs d'explorer uniquement une petite portion du voisinage de la solution courante. Des implémentations asynchrones peuvent également être exploitées en lançant différents processus de *recherche tabou* séparés sur différents processeurs et en les faisant communiquer au travers d'un espace mémoire partagé. Dès que la meilleure solution globale est améliorée par l'un des processus, la nouvelle meilleure solution peut être mémorisée sur la mémoire partagée. À contrario, un processus n'arrivant pas à améliorer sa meilleure solution sur une longue période pourrait utiliser la meilleure solution globale présente dans l'espace mémoire partagé pour redémarrer son processus de recherche. Des dispositifs similaires sont possibles pour paralléliser l'algorithme du *recuit simulé*. Concernant les algorithmes génétiques, ils sont faciles à paralléliser car ils suivent nativement différents chemins de recherche. La méthode *GRASP* peut également profiter du parallélisme grâce à une génération parallèle de nombreuses solutions à la phase de construction. Au final, des approches hybrides sont possibles. Par exemple, on peut lancer différents processus de *recherche tabou* concurrents sur différents processeurs pour combiner les meilleures solutions trouvées de deux processus et utiliser cette nouvelle solution croisée comme solution de départ sur un nouveau processeur.

6.3 Principe de maintien de solution dans un système dynamique

Nous étudions ici certains procédés [276, 22] liés au cadre des problèmes dynamiques de satisfaction de contraintes (section 5.1.3) susceptibles d'être transposés à notre problème dynamique d'optimisation sous contraintes (section 5.1.4).

Bien que la recherche se soit focalisée très longtemps sur des problèmes statiques, la plupart des problèmes de RO ou d'IA sont en réalité des problèmes dynamiques, c'est à dire des problèmes où l'ensemble

des variables et des contraintes à considérer pendant la phase de résolution évolue du fait de changements (événements) provenant soit de l'environnement, soit de l'utilisateur ou plus généralement d'autres agents.

À titre d'exemples, ces changements peuvent intervenir :

- au niveau de l'*environnement* : l'état physique du système permettant de réaliser les actions prévues évolue (pannes, incidents, ressources non disponibles, effets des actions, ...), ce qui rend leur exécution rarement conforme aux prévisions ;
- au niveau de l'*utilisateur* : pour des problèmes de conception interactive, l'utilisateur travaille par explorations successives de spécifications différentes pour évaluer les implications liées à tel ou tel choix ;
- au niveau des *agents* : les décisions d'un agent dans un système distribué modifient les spécifications des problèmes traités par d'autres agents.

Les systèmes d'aide à la décision font partie des nombreux problèmes soumis à des perturbations [86] où l'utilisateur peut commander le système en lui fournissant de nouvelles données et où le solveur informe l'utilisateur en retour.

6.3.1 Besoins utilisateurs

Concernant l'utilisateur, l'efficacité de la recherche et la stabilité des solutions successives sont deux besoins essentiels :

- le besoin de mécanismes de recherche efficaces peut être plus fort dans le cadre de problèmes dynamiques où le temps disponible pour produire une nouvelle solution est souvent limité.
- la stabilité des solutions successives peut être importante au cas où un certain travail a pu être engagé sur la base de la solution précédente. La stabilité est une propriété qui lie les solutions successives produites par un système dynamique. Une solution est stable par rapport à la solution précédente si le passage d'une solution à l'autre se fait par de petits changements [86].

6.3.2 Réparation de la solution précédente

La méthode proposée par [192] s'attache à réparer une affectation complète issue de la recherche précédente plutôt qu'à reconstruire une solution à partir de zéro. Une heuristique de réparation ou de minimisation des conflits consiste à choisir une variable intervenant dans une contrainte insatisfaite et à choisir, pour cette variable, une valeur qui minimise le nombre de contraintes insatisfaites. Ces heuristiques peuvent être utilisées aussi bien dans le cadre d'une recherche systématique (section 5.2.1) ou non systématique (section 5.2.2). Cette méthode donne de bons résultats à la fois en termes d'efficacités et de stabilité, sur des problèmes sous-contraints. Néanmoins, pour fournir une solution proche de la solution précédente, des techniques ont du être développées pour répondre notamment à l'objectif de minimisation du nombre de variables dont l'affectation est modifiée [21]. Notons que le critère de distance entre deux solutions peut être agrégé avec d'autres critères de type *coût* propres au contexte.

6.4 Conclusion

Les SIAD changent la manière de prendre les décisions face à une information qui a tendance à se globaliser, des problèmes du plus en plus difficiles à résoudre et des délais de réflexion qui se réduisent. Ces systèmes représentent désormais des dispositifs de collaboration particulièrement adaptés entre l'homme et la machine pour accompagner des processus complexes devenus très souvent collectifs. Ils reposent en partie sur des techniques issues du domaine des mathématiques appliquées comme l'optimisation combinatoire pour aider efficacement les décideurs dans leurs tâches les plus ardues, notamment face à des problèmes peu ou mal structurés. Le maintien de solution dans un cadre d'aide à la décision est un problème supplémentaire qui a été abordé assez tardivement par les communautés scientifiques malgré une profusion de cas

disponibles dans nos quotidiens. Il pose une double difficulté : efficacité accrue de la résolution et stabilité des solutions proposées. Dans ce contexte, La technique qui consiste à réutiliser une solution précédente sur un problème dynamique semble être une voie incontournable pour répondre à ces deux objectifs. Enfin, en résolution de problèmes combinatoires, garantir la prédictibilité des temps de calculs et la qualité des solutions disponibles à chaque instant est particulièrement difficile à assurer avec les algorithmes classiques, complets ou incomplets. L'ensemble de ces difficultés fait probablement que peu d'applications d'aide à la décision développées autour des problèmes combinatoires sur de grosses instances soient réellement des applications temps réel.

7

Modélisation urbaine

Compte tenu de la puissance de calcul des ordinateurs actuels, le recours à la simulation est un moyen crédible de nos jours pour s'attaquer à des problématiques complexes auxquelles nos sociétés modernes sont confrontées. Or, toute simulation doit s'appuyer sur un modèle, c'est à dire une représentation simplifiée d'une réalité qui lui donne du sens et permet de mieux la comprendre [123].

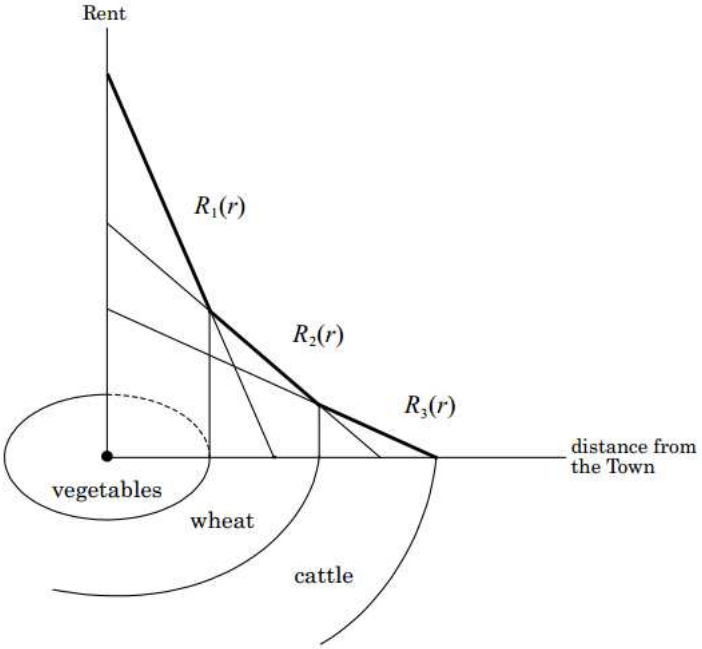
En matière d'environnements urbains, de nombreux travaux ont été proposés ces dernières décennies pour mieux comprendre l'organisation des villes. C'est le cas notamment des modèles d'économie spatiale, d'écologie urbaine ou encore d'interaction spatiale. Plus récemment, on a assisté à l'émergence de l'approche systémique et individu-centrée annoncée comme une évolution majeure. Ces derniers modèles sont réputés pouvoir représenter des systèmes complexes dans leur globalité en se basant sur le concept d'émergence provenant des interactions entre les individus (emplacements, habitants, emplois, résidences, entreprises, routes, ...) qui composent le système.

Dans ce chapitre, nous présentons un état de l'art de la modélisation urbaine restreint à des thèmes proches de notre problématique. Pour positionner nos travaux dans le panorama de la recherche actuelle, nous détaillons plus particulièrement trois approches représentées par les projets de recherche *UrbanSim* et *GeOpenSim* d'une part, et par la plateforme commerciale *CommunityViz* d'autre part.

En effet, de nombreux projets de recherche relatifs aux problématiques urbaines sont initiés sur les différents continents. Nombre d'entre eux s'intéressent à la fois aux problématiques du transport et à celles de l'occupation du sol, le but étant de mieux évaluer l'impact des choix actuels sur le long terme. Ces modèles sont regroupés sous l'abréviation LUTI (Land Use and Transport Interaction model). Dans ce contexte, nous avons souhaité étudier le projet de recherche *UrbanSim* mis au point aux États-Unis par l'équipe de Paul WADDEL et ayant une forte orientation pour l'aménagement des espaces urbains. Ce modèle dynamique a su s'imposer ces dernières années, y compris au sein de certains projets de recherche européens. Mais ce projet nécessite beaucoup de données en entrée. Or, beaucoup d'informations utiles peuvent être exploitées à partir d'une simple photographie. C'est l'approche originale qui a été retenue par le projet de recherche *GeOpenSim* qui a suscité notre curiosité et que nous décrivons brièvement. D'un autre côté, on trouve la plateforme commercialisée sous le nom de *CommunityViz* qui se présente comme un outil de nouvelle génération pour la planification urbaine : voyons ensemble si cet outil est adapté à nos besoins et quelles sont les réponses innovantes qu'il est en mesure de délivrer aux urbanistes et décideurs ?

Indépendamment des modèles spécialisés relatifs à l'urbanisme, la famille des problèmes génériques de positionnement révèle de très fortes similitudes par rapport à notre problématique qui consiste à positionner au mieux des éléments urbains (habitations, écoles, commerces, ...) sur chaque cellule d'une grille (emplacements libres), chaque élément devant être associé à une et une seule cellule. Parmi les nombreux problèmes de positionnement référencés, nous introduisons plus particulièrement le problème d'affectation

FIGURE 7.1 – Profit de la rente foncière et anneaux de Von THÜNEN (pour 3 anneaux). D'après J.H. VON THÜNEN et al. [279].



quadratique représentatif des problèmes d'optimisation fortement combinatoire. L'étude de ce problème de référence nous apportera peut-être les derniers éléments de réponse utiles à notre modèle spécifique de pré-programmation urbaine.

7.1 Représentations historiques de la ville

Les premières représentations de la ville initiées par les économistes remontent au début du 19ème siècle. Elles considèrent généralement la proximité urbaine comme source de profits, ces profits pouvant être maximisés en choisissant une localisation optimale pour chaque type d'activité.

Ces travaux pouvaient s'intéresser aux critères de choix d'implantation d'une industrie, aux rapports entre localisation et rente foncière en fonction de la distance au centre ville, en encore à la répartition des villes et des activités au sein d'un espace régional.

7.1.1 Les modèles d'économie spatiale

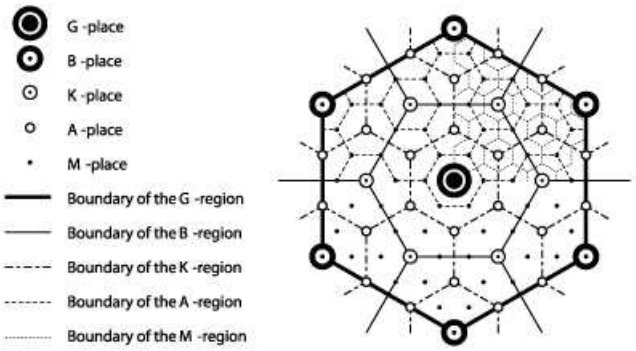
Les premiers modèles d'urbanisation cherchent à étudier et à prévoir l'occupation du sol entre différents types d'activités (résidentielles, économiques, équipements, ...). Ils ont une portée essentiellement théorique et leur apport à l'urbanisme opérationnel reste généralement limité.

Dès 1820, Von THÜNEN cherche à expliquer la localisation des activités agricoles autour des villes allemandes. Grâce à la formulation de plusieurs hypothèses sur les bases d'une ville théorique et d'un espace isotrope[1], il a démontré que la production d'une denrée n'est rentable qu'à une distance donnée du marché compte tenu de son coût de transport et du coût de la terre (rente foncière) [279]. Ainsi, plus le coût de transport d'un produit est élevé, plus le produit sera proche du marché (cf. figure 7.1).

En 1933, W. CHRISTALLER [45] avec sa théorie des lieux centraux (cf. figure 7.2) tente d'expliquer la taille, le nombre et la répartition hiérarchique et géographique des villes au sein d'un espace régional.

[1]Qui représente les mêmes caractéristiques physiques dans toutes les directions, en tout point homogène.

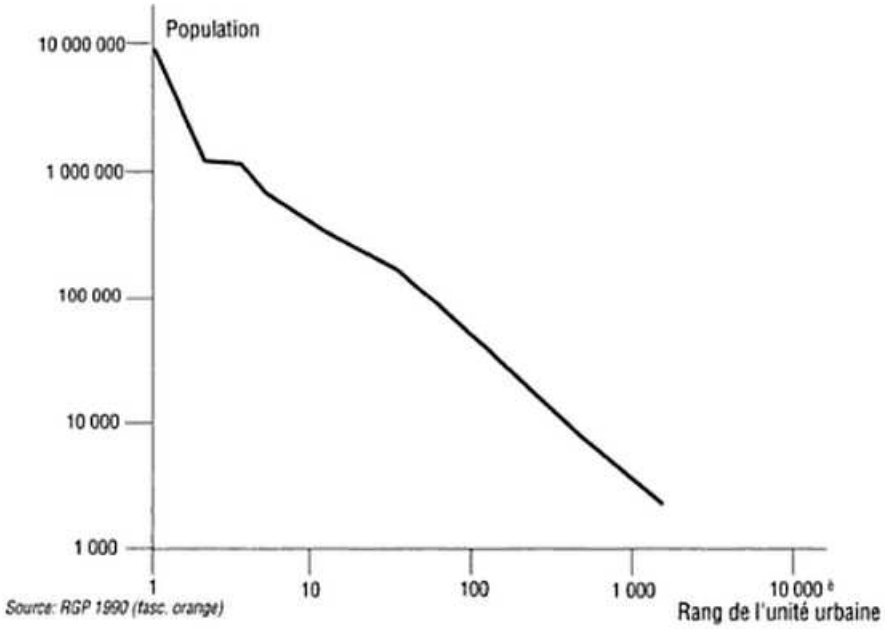
FIGURE 7.2 – Modèle de CHRISTALLER : théorie des lieux centraux. D'après W. CHRISTALLER [45].



Il montre qu'une région s'organise hiérarchiquement en fonction de l'offre en biens qu'elle propose sur la base d'hypothèses simplistes (accessibilité et niveaux de revenus égaux en tout point, etc.).

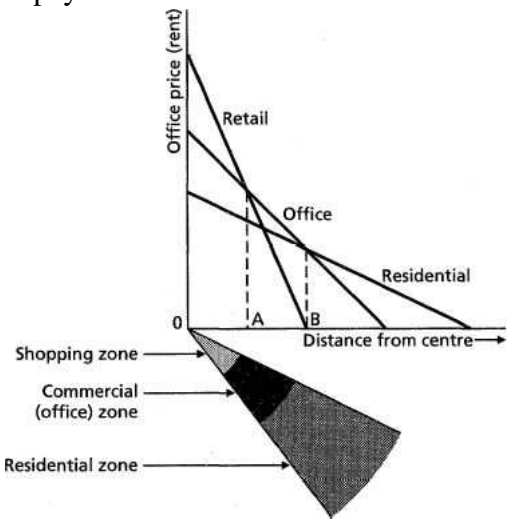
En partant des travaux de A. LÖSCH (1940) [175] qui utilise des méthodes de découpage de l'espace en zones homogènes pour faire des analyses spatiales de phénomènes urbains, K.G. ZIPF propose la loi rang-taille en 1949 [292]. Cette loi considère que la taille des villes est distribuée de manière inverse à leur rang, indépendamment de l'espace et du temps (cf. figure 7.3).

FIGURE 7.3 – Modèle de ZIPF : relation rang-taille des unités urbaines françaises en 1990. D'après FABRIÈS-VERFAILLIE et al. [89].



En 1964, W. ALONSO [6] donne naissance à l'économie urbaine en élaborant un premier modèle basé sur la théorie de la croissance urbaine. Ces modèles liés à l'étude des valeurs foncières veulent expliquer le comportement des ménages et des activités économiques en cherchant un compromis entre proximité, pénibilité du trajet, disponibilité du terrain et leur prix. Dans sa théorie, l'organisation spatiale de l'utilisation du sol est déterminée par la valeur foncière, elle-même liée aux coûts de transport. Chaque activité est en mesure de payer une rente en fonction de la distance au centre, les différentes localisations entre activités étant interdépendantes (cf. figure 7.4).

FIGURE 7.4 – Modèle de rente à payer en fonction de la distance au centre. D'après W. ALONSO [6].

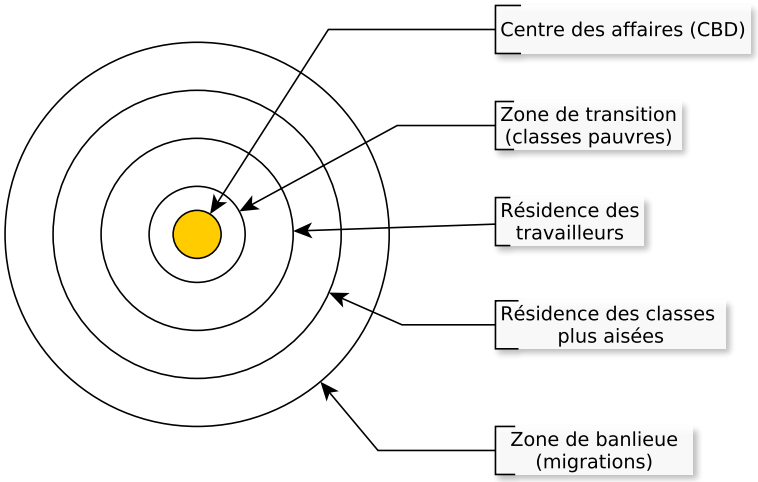


7.1.2 Les modèles de l'écologie urbaine

L'écologie urbaine introduite par l'école de Chicago [120] consiste à étudier les interactions entre les êtres vivants et la ville. Elle marque les débuts d'une analyse des espaces urbains par les sciences sociales en proposant des modèles explicatifs de répartition des populations en fonction de leurs caractéristiques sociales, familiales, ethniques, etc. Dans cette lignée, on recense trois principaux modèles complémentaires qui décrivent la ville sous forme d'aires suivant différents critères.

Le modèle de BURGESS [37] (1925) étudie les itinéraires résidentiels socio-spatiaux et prend la forme d'un schéma concentrique pour d'écrire l'organisation d'un espace urbain (cf. figure 7.5).

FIGURE 7.5 – Modèle radioconcentrique de BURGESS. D'après E.W. BURGESS [37].



Le modèle de HOYT [137] (1939) considère que des conditions particulières ou des lignes de communication favorables ont un effet structurant sur les zones voisines qui deviennent plus attractives. À partir de cette idée, la ville est structurée en arcs de cercles et en secteurs (cf. figure 7.6).

En 1945, HARRIS et ULLMAN [129] s'appuient sur une organisation de l'agglomération constituée de noyaux, chaque noyau étant caractérisé par une fonction et articulé par des axes de transport (cf. figure 7.7). Dans leur modèle, certaines fonctions se repoussent alors que d'autres s'attirent ou dépendent des caractéristiques du site.

FIGURE 7.6 – Modèle de HOYT sur la théorie des secteurs : (1) Centre des affaires, (2) Industries et entrepôts, (3) Résidences des classes pauvres, (4) Résidences des classes moyennes, (5) Résidences des classes aisées. D'après H. HOYT [137].

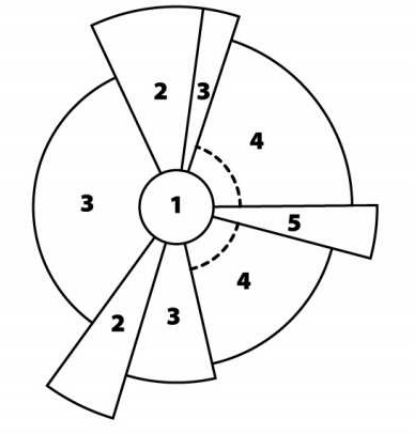
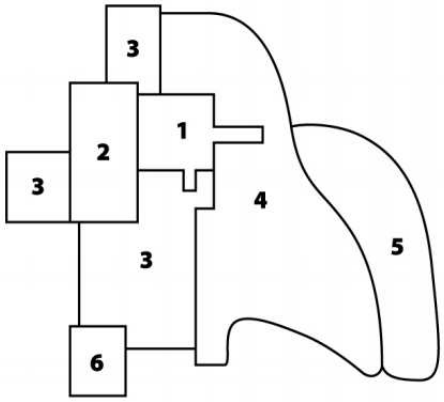


FIGURE 7.7 – La théorie des centres multiples : (1) Centre des affaires, (2) Industries légères et entrepôts, (3) Résidences des classes pauvres, (4) Résidences des classes moyennes, (5) Résidences des classes aisées, (6) Industries lourdes. D'après C.D. HARRIS et E.L. ULLMAN [129].



7.1.3 Les modèles d'interaction spatiale

La notion d'interaction spatiale est souvent définie comme l'étude de la décroissance des flux avec la distance, la distance participant notamment à la structuration de l'espace géographique par des champs de force [224].

Cette approche, ramenée par analogie aux lois de la gravitation universelle, a donné lieu aux modèles déterministes de E.G. RAVENSTEIN [225] qui remontent à 1889. En général, ces modèles gravitaires font dépendre le volume d'interaction entre deux lieux de la masse des lieux émetteurs et récepteurs ainsi que de l'inverse du carré de la distance qui les sépare.

Les modèles gravitaires de STEWART [259] et de ZIPF [293] sont conformes à la théorie de la gravité de NEWTON : l'interaction entre deux unités géographiques est proportionnelle au produit de leurs masses respectives (population, richesse, etc.) et inversement proportionnelle à la distance qui les sépare. Par ailleurs [224], ZIPF a proposé d'appeler « *loi du moindre effort* » la propension universelle à couper au plus court et à aller au plus proche : ce qui revient à organiser les activités et les déplacements en fonction de la distance.

Les modèles de REILLY [229] et HUFF [139] s'attachent quant à eux à déterminer les aires de marché théoriques d'un ensemble de lieux centraux. Ils cherchent à décrire les lieux appartenant à une zone de marché en exploitant les relations entre ces lieux.

Comme le souligne Denise PUMAIN dans son "essai sur la distance et l'espace géographique" [224], en référence au cartographe américain Waldo TOBLER : « *tout interagit avec tout, mais la première loi de la géographie est que deux choses proches ont plus de chances d'entrer en interaction que deux choses éloignées* ».

Bien que la majorité des modèles étudiés dans les sections précédentes n'apportent que de faibles justifications théoriques, souvent dépassées et incapables à elles seules de décrire les situations actuelles, ils constituent une base solide pour les modèles qui suivront.

7.2 Vers une approche systémique et individu-centrée

J. DE ROSNAY définit un système comme "*un ensemble d'éléments en interaction dynamique, organisés en fonction d'un but*" [65]. Ainsi, la majorité des problèmes actuels peuvent être considérés comme des systèmes complexes dans la mesure où ils font intervenir un grand nombre de variables connectées entre elles, ils sont difficiles à comprendre dans leur globalité et peuvent être représentés de différentes façons.

L'approche *analytique* et *sectorielle* à partir de laquelle les systèmes complexes ont longtemps été étudiés et décomposés en autant de champs que possible (économie, habitat, transport, etc.) est nécessairement réductionniste et montre d'importantes limites [8]. Cette approche consistant à réduire la nature complexe des choses en la décomposant mène à des solutions partielles qui ne traitent qu'un aspect du problème. Et ces solutions fragmentaires ont souvent des conséquences indirectes non maîtrisées sur d'autres aspects du problème. Au contraire, l'approche *systémique* proposée dès 1968 par L. VON BERTALANFFY [278] considère un système complexe comme un ensemble de systèmes emboîtés les uns dans les autres et orientés dans une finalité commune [154]. Les parties du système étudié ne peuvent être comprises si elles sont considérées isolément. Elles sont dynamiquement reliées entre elles dans une interaction et une interdépendance permanentes [95]. Cette démarche envisage tout système complexe comme un ensemble d'éléments faisant système et encourage à un travail transdisciplinaire [39]. L'approche systémique permet d'appréhender un problème dans sa globalité pour proposer des solutions qui prennent en compte l'ensemble des effets, "*un tout ne se réduisant pas à la somme de ses parties constitutives*" [197].

La ville est faite de multiples composants, articulés entre eux au sein de sous-systèmes urbains en interactions dynamiques, ces interactions étant à l'origine du comportement du système dans le temps. La ville peut donc être vue comme un système complexe. Dès lors, l'approche systémique appliquée à la ville apparaît pertinente pour la considérer comme « un tout » et rendre compte de sa complexité et de son auto-organisation [223]. Ainsi, avec la parution de *Urban Dynamics* [96], J. FORRESTER est le premier à proposer un modèle de dynamique urbaine destiné à la simulation du développement à très long terme et à l'exploration des conséquences de politiques alternatives. Cette étude qui rompt avec les modèles précédents a notamment permis de qualifier les systèmes urbains comme des systèmes contre intuitifs et extrêmement robustes (peu sensibles aux valeurs des paramètres ou variables d'entrée du modèle).

A partir des années 1970, les termes quantification, modélisation, approche systémique et analyse spatiale sont devenus des mots clés pour tout géographe. C'est alors que les travaux de modélisation se généralisent et permettent la construction de scénarios, la discussion et la confrontation d'idées. Avec les progrès techniques, on accède progressivement aux possibilités de traitement des données spatialisées. Cette évolution permet le couplage des modèles avec des systèmes d'informations géographiques (SIG) et le développement des modèles individu-centrés essentiellement composés des automates cellulaires[2] et des systèmes multi-agents[3]. Les modèles centrés individus permettent de prendre en compte un grand nombre d'individus (i.e. cellules ou agents) en désagrégeant les comportements de chacun d'entre eux. Cette technique liée à

[2]Un réseau d'automates cellulaires (AC) se présente comme "*un système de cellules interagissant localement de manière simple et manifestant un comportement global complexe*"(Stephen WOLFRAM).

[3]Système composé d'un environnement, d'un ensemble d'objets passifs localisés et d'un ensemble d'agents actifs situés capables de percevoir, produire, consommer, transformer et manipuler les objets (Jacques FERBER).

une dimension stochastique enrichit nettement la modélisation et prend le parti pris de l'émergence, l'un des concepts les plus intéressants des modèles systémiques [39].

Il est indéniable que la modélisation à base d'agents (Agent-based modelling) a connu une très forte adhésion pour la simulation sociale. Son succès peut s'expliquer en raison d'une vision du monde qui suggère que les systèmes complexes émergent de la base, sont fortement décentralisés, et sont composés d'une multitude d'objets hétérogènes. Néanmoins, avant d'adopter ce paradigme, il convient de prendre en compte les nombreux défis que ce type de modélisation soulève encore [59] :

- construire le modèle au bon niveau de détail pour bien servir le but recherché,
- proposer un modèle qui puisse être généralisable (et non spécifique à un problème donné),
- assurer la robustesse du modèle dans différentes situations, notamment avec des données différentes,
- mettre au point un protocole pour vérifier et valider le modèle,
- permettre aux autres intervenants de comprendre le modèle pour favoriser son adoption,
- ....

Quelle que soit la technique employée, la modélisation urbaine a su se développer dans de nombreux domaines et conduira au développement des modèles LUTI (Land Use and Transport Interaction) qui permettent de modéliser les interactions entre les problématiques du transport et celles de l'occupation et de l'usage des sols.

7.3 La modélisation LUTI

Les logiciels professionnels de type "*transport-urbanisation*" ou "*transport-occupation du sol*" plus connus sous l'abréviation modèle LUTI (*Land Use Transport Interaction model*) représentent encore un marché balbutiant. En fait, l'intérêt d'associer ces deux types de modélisation (transport et occupation des sols) n'est apparu que depuis une dizaine d'années [61] et la plupart des modèles LUTI actuels sont en réalité des modèles "connectés" résultant d'un couplage à posteriori d'un modèle de trafic et d'un modèle d'urbanisation. En France, plusieurs projets de recherche de ce type ont vu le jour sous l'impulsion du PREDIT[4]. Par exemple, on peut citer les projets *SIMAURIF*[5] en Île-de-France, *SIMBAD*[6] à Lyon, *MIRO*[7] ou encore *MOBISIM*[8]. On relève également des initiatives dans le secteur privé avec VINCI qui a proposé le modèle *PIRANDELLO*[9]. On peut retrouver dans le livre "*Modéliser la ville*" [8] une description détaillée de ces différents projets.

Les modèles LUTI simulent les effets d'une nouvelle infrastructure de transport sur l'occupation des sols à moyen ou long terme :

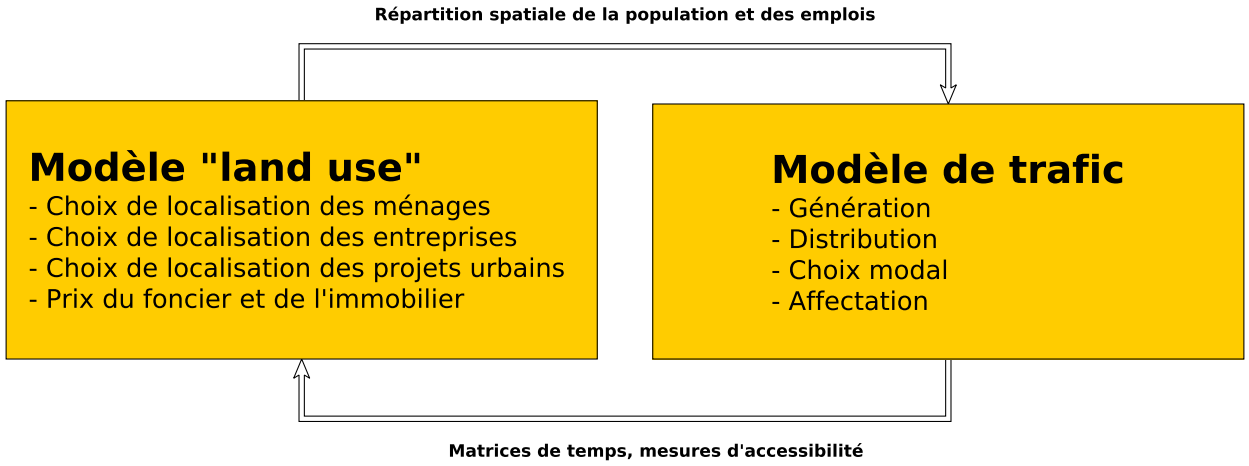
- localisation des ménages ;
- localisation des entreprises ;
- effets sur le prix du foncier et de l'immobilier.

Une boucle rétroactive se met en place entre le modèle de transport et le modèle d'urbanisation (voir Figure : 7.8). Sous l'effet des nouvelles infrastructures de transport, les prix du foncier et de l'immobilier évoluent et la répartition spatiale des populations et des emplois est également affectée. Cette nouvelle répartition génère des déplacements qui sont réaffectés sur le réseau de transport. Les disponibilités foncières et leurs prix sont à nouveau impactés ainsi que les réseaux de transport, etc. En général, les effets de la boucle de rétroaction sont évalués annuellement sur une période pouvant aller de 20 à 30 ans.

Un modèle LUTI peut apporter des réponses à des questions du type :

[4] PREDIT : Programme de Recherche et d'Innovation dans les Transports Terrestres (http ://www.predit.prd.fr/predit4/).
[5] http://isidoredd.documentation.developpement-durable.gouv.fr/documents/dri/RMT08-010.pdf
[6] http://simbad.let.fr/
[7] http://isidoredd.documentation.developpement-durable.gouv.fr/documents/dri/RMT07-013.pdf
[8] http://www.mobisim.org/
[9] http://www.etudesfoncieres.fr/articles/article-pirandello-un-modele-urbain-global-137-gratuit.pdf

FIGURE 7.8 – Schéma simplifié d'un modèle dynamique de type LUTI. D'après le rapport final sur la première phase d'ULTISIM [61].



- quelles sont les plus-values foncières et immobilières générées par une nouvelle infrastructure de transport en commun à proximité d'une gare ?
- quel est l'impact d'une politique de transport particulière sur la relocalisation des ménages et des emplois ou sur les émissions de CO_2 ?
- quels sont les effets d'un péage autour du centre ville sur le développement économique régional et local ? Va-t-il favoriser un étalement urbain ou une densification ?
- etc.

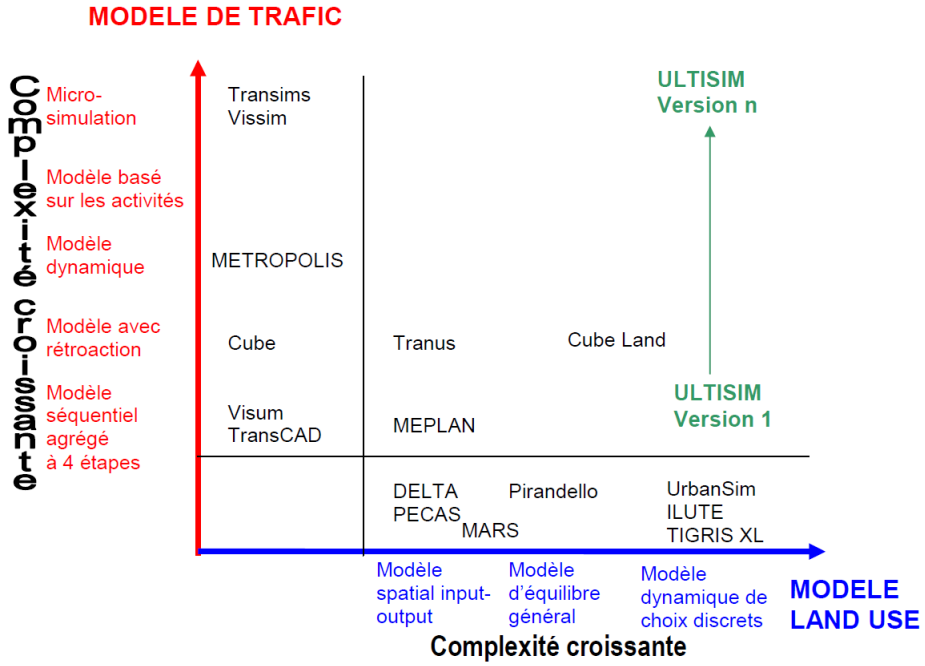
Cependant, un modèle macro-économique externe est nécessaire pour introduire par exemple la part de création nette d'emplois en faisant des hypothèses de totaux par année de simulation. Pour plus de précisions sur les modèles existants, on pourra se référer au rapport final du projet ULTISIM[10] [61] qui propose une étude de six modèles LUTI opérationnels, à savoir (par ordre alphabétique) :

- CUBE LAND ;
- DELTA (Development, Employment, status and communting, Location and property market, Transition and growth, and Area quality) ;
- MARS (Metropolitan Activity Relocation Simulator) ;
- OPUS/UrbanSim ;
- PECAS (Production, Exchange and Consumption Allocation System) ;
- TRANUS ;

La figure 7.9 reprend pour chaque modèle LUTI son niveau de complexité et son degré d'implication comme modèle d'urbanisation ou de transport. Compte tenu de son positionnement (Land Use Model) et de son niveau de complexité (modèle dynamique à choix discrets), nous proposons d'étudier plus en détail la plateforme *UrbanSim* dans la section suivante ; ceci pour mieux appréhender l'état de l'art en la matière à partir d'un outil réputé et faisant référence dans son domaine. De plus, nous souhaitons évaluer dans quelle mesure un tel modèle pourrait nous être utile pour répondre à notre problématique. En effet, bien que notre projet ne soit pas lié à l'aspect dynamique évoqué ici, la dimension temporelle n'intervenant pas dans notre cas, nous faisons l'hypothèse que nous pourrions utiliser le modèle *UrbanSim* pour générer une ville de toute pièce. On partirait d'une ville initialement vide sans habitant ni bâtiment. A chaque année, il

[10] Le projet de recherche *ULTISIM* (Urban Land Transport Integrated SIMulation Model) a pour ambition de proposer un modèle intégré transport-urbanisme au niveau européen. Ce projet de recherche financé dans le cadre du PREDIT se décompose en deux phases : une phase de conception et une phase de développement. La première phase du projet prévue sur 20 mois (et achevée en juin 2011) consiste à mener une réflexion approfondie sur les données d'entrée et sur la méthodologie de développement pour aboutir à terme à un modèle LUTI opérationnel en Île-de-France.

FIGURE 7.9 – Positionnement des modèles LUTI. D'après le rapport final sur la première phase d'ULTISIM [61].



faudrait injecter dans notre modèle un certain nombre d'éléments urbains (habitants, emplois, habitations, entreprises, parcelles, routes, ...) et laisser le modèle œuvrer pour répartir ces éléments au mieux sur le territoire. Le processus se poursuivrait d'année en année jusqu'à ce que tous les éléments prévus dans le modèle urbain de pré-programmation (vu comme un modèle macro-économique externe) soient intégrés. La ville ainsi constituée correspondrait au résultat d'une simulation, le nombre d'années utilisées pour les opérations n'ayant que peu d'importance dans notre situation (car nous aurions détourné le modèle de sa fonction originale) ...

7.4 Trois modèles urbains passés au crible

7.4.1 Le projet *UrbanSim*

UrbanSim [283] est un modèle d'urbanisation (land-use model) basé sur la théorie des modèles de choix discrets[11] et sur le principe de la microsimulation[12]. *UrbanSim* est apparu aux États-Unis en 1999 pour répondre aux nouvelles exigences de la loi, à savoir l'obligation de prendre en compte dans tout processus de planification régional les dimensions relatives à l'urbanisation, aux transports et à l'environnement. Il s'agit d'un logiciel de simulation complet qui repose sur une approche à base de modèles pour la planification et l'analyse du développement urbain, en prenant en compte les différentes interactions entre l'utilisation des espaces (terrains), les transports ainsi que les politiques publiques. Il est destiné aux organismes de planification métropolitaine et autres groupements qui ont besoin d'intégrer des modèles de transport existants avec une nouvelle utilisation des espaces dans le but de faire des prévisions et des analyses de capacités.

Il a vu le jour sous l'impulsion de P. WADDELL au sein du CUSPA[13] de l'Université de Washington à Seattle. *UrbanSim* est l'aboutissement de nombreuses recherches commencées dès les années 1960 en

[11] La théorie des choix discrets a pour objet de décrire le comportement de choix d'un individu (ou d'un groupe d'individus agissant indépendamment les uns des autres) en présence d'actions mutuellement exclusives [28].

[12] L'idée maîtresse de la microsimulation est que le meilleur moyen de simuler un système consiste souvent à modéliser et à simuler les actions et les interactions de ses unités de plus petite échelle et d'obtenir les macrorésultats par agrégation [207].

[13] Center for Urban Simulation and Policy Analysis

Amérique latine puis aux États-Unis. P. WADDELL a su tirer parti des autres logiciels existants à cette époque pour concevoir un système qui intègre toutes les fonctionnalités d'un logiciel de prévision de l'occupation des sols, basé sur un modèle de prix du foncier et des modèles de choix discrets de localisation des ménages et des entreprises. Avec ce logiciel, chaque ménage, chaque emploi et chaque unité géographique (cellules paramétrables de 500 mètres de côté par défaut) sont représentés dans le modèle, chacun avec ses caractéristiques et ses comportements.

7.4.1.1 Motivations de base

Tout les aspects d'une agglomération sont interconnectés et interdépendants. Une action sur un secteur influence automatiquement les autres secteurs à des degrés différents et en fonction des autres actions opérées au même moment. La coordination des différents événements constitue donc un élément important qui n'est pas souvent pris en considération et qui peut aboutir parfois à des comportements inattendus. De plus, il existe une grande variété d'intervenants différents (urbanistes, organismes publics, citoyens et associations, etc.), qui, chacun à son niveau, doit pouvoir constater les conséquences et les alternatives possibles des choix politiques et investissements. L'outil doit faciliter les délibérations publiques parfois houleuses relatives à l'aménagement du territoire, aux transports ou à l'environnement en indiquant les conséquences et alternatives possibles sur les effets à longs termes sur l'environnement et les considérations d'équité. Enfin, il doit permettre aux différentes communautés (au sens de l'agglomération) d'avoir une vision globale pour coordonner leurs actions.

7.4.1.2 Approche

La conception d'*UrbanSim* prend en compte les expériences du passé en terme de modélisation de l'urbanisation. Cet environnement s'appuie sur un modèle dynamique (où la dimension du temps intervient) qui reproduit l'évolution de l'occupation du sol au cours de la période simulée. La structure de données d'*UrbanSim* permet de prendre en compte des données d'occupation du sol à un niveau infracommunal[14] et donc de dépasser des approches classiques trop simplificatrices (découpage concentrique en trois zones ou découpage par agrégation de communes). Les différents modules d'*UrbanSim* requièrent des données spécifiques. Les données d'entrée sont composées :

- des données de l'année initiale ;
- des données de prévision macro-économiques ;
- des données sur les transports issues du modèle de trafic (temps et coûts de parcours par mode, etc.) ;
- les scénarios de politiques d'urbanisation tenant compte des schémas d'aménagement locaux et régionaux.

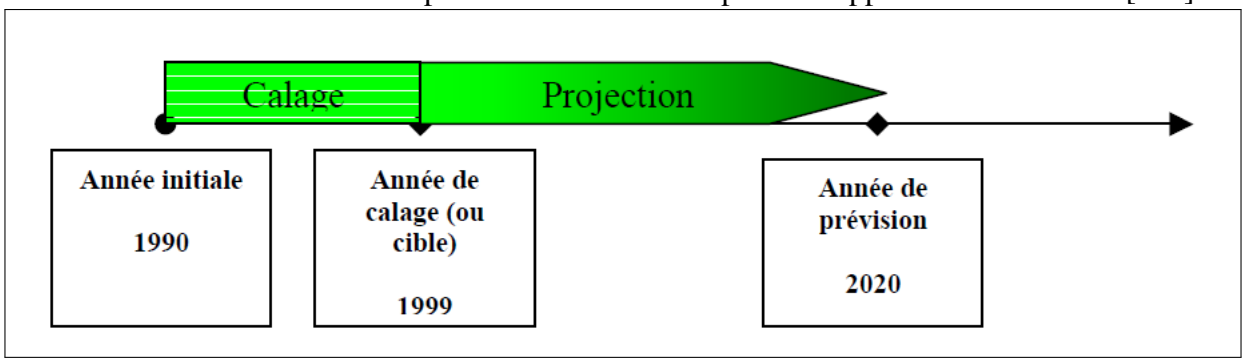
Les différents modules sont interdépendants car les sorties de l'un sont les entrées des autres. Le déroulement des calculs est séquentiel et l'ordre peut être paramétré par l'utilisateur dans un fichier XML. Le logiciel est conçu comme un ensemble de composants reflétant les choix des agents économiques (ménages, entreprises) et leurs interactions avec le marché de l'immobilier. On retrouve par exemple :

- les choix de localisation résidentielle des ménages ;
- les choix de localisation des emplois ;
- les choix de développement immobilier.

Les modules de localisation sont basés sur un quadrillage fin de l'aire d'étude et les choix sont relatifs à chaque cellule de ce quadrillage. Le secteur d'étude doit être découpé en mailles ou cellules carrées les plus fines possibles, et les données d'entrée sont à recueillir dans ce maillage, pour l'année initiale et l'année de calage. Le calage d'*UrbanSim* consiste à estimer les paramètres du modèle à partir de données désagrégées sur une période constituée de deux années pour lesquelles il existe le maximum de données. Le modèle sera

[14] à l'intérieur des limites communales

FIGURE 7.10 – Cours du temps dans *UrbanSim*. D'après le Rapport 1 - SIMAURIF [246]



ensuite appliqué, avec les mêmes paramètres, à un horizon futur selon différents scénarios à définir (voir figure 7.10).

Certains composants sont capables de simuler la mobilité des ménages et des emplois. En parallèle, il existe des composants basés sur des données agrégées, comme les modules de transitions économique et démographique. Ces modules non spatiaux sont liés aux données macro-économiques exogènes et permettent de caler le modèle au niveau global. Un autre module permet de simuler le prix du foncier. Les versions d'*UrbanSim* les plus récentes proposent des modules complémentaires (choix de localisation des entreprises, choix de développement de projets, etc). Les différents modules s'exécutent sur un pas d'un an, pendant toute la période de simulation (par exemple une période de vingt ans). Par ailleurs, *UrbanSim* utilise une structure hiérarchique pour organiser les différentes simulations appelées "scénarios". *UrbanSim* est prévu pour simuler et évaluer les effets potentiels de scénarios multiples. Dans ce contexte, un scénario correspond à une combinaison de données en entrée et d'hypothèses, en incluant les hypothèses macroéconomiques au regard de l'augmentation de la population et de l'emploi pour la zone étudiée. On considère que la configuration des moyens de transport est connue pour les prochaines années et que les plans généraux des juridictions locales qui vont définir les types de développement sont disponibles pour chaque localité. Une simulation *UrbanSim* consiste à faire tourner le moteur *UrbanSim* sur un "scénario". Les résultats de la simulation sont alors stockés dans une base de données unique (Output Database). Il existe un moyen dans *UrbanSim* de prendre en compte des événements exceptionnels prévus dans le futur. Il suffit pour cela de définir dans les fichiers de type « events » les nombres d'emplois, de ménages, le changement de type urbain et l'année pour les cellules où se produisent ces événements. Enfin, le module d'export de données permet le regroupement, l'agrégation et l'export des résultats dans des fichiers externes destinés à l'analyse et à l'importation dans un système d'information géographique.

7.4.1.3 Un système basé sur les modèles

Les modèles d'*UrbanSim* interviennent à différents niveaux :

- simulent les décisions et choix qui impactent le développement urbain (mobilité, choix d'implantation des habitations et des zones d'affaires, choix des constructeurs) ;
- prennent explicitement en compte les terrains, les structures (maisons d'habitation et bâtiments commerciaux) ainsi que les occupants (ménages, commerces) ;
- simulent le développement urbain comme un processus dynamique dans le temps et l'espace ;
- simulent le prix des terrains en fonction de l'offre (espaces libres, nouvelles constructions ou rénovations) et de la demande (emplacements préférés pour les maisons d'habitations ou les commerces) ;
- tiennent compte des choix politiques gouvernementaux et évaluent les impacts politiques en modélisant les réactions du marché ;
- sont basés sur la théorie aléatoire pour impacter les composants ;
- sont conçus pour travailler sur une représentation de haut niveau de l'espace et des activités avec un

système de zones identique à celui des déplacements ;
- sont adaptés aux besoins de développement ou de réaménagement, en travaillant au niveau du détail des parcelles.

7.4.1.4 Une plateforme ouverte d'intégration

Une plateforme ouverte a été développée pour y intégrer *UrbanSim*. Il s'agit de la plateforme *OPUS*[15] qui permet une intégration d'extensions (plugin). *UrbanSim* constitue la pièce centrale de cette plateforme. Après un premier développement en *Java*, l'application a été totalement réécrite en *Python* pour faciliter l'intégration et son ouverture. Les développements exploitent le paradigme de la programmation orientée objet pour offrir un maximum de flexibilité. L'application dispose d'une interface graphique écrite en *PyQt4*. Elle peut s'interfacer avec les principales bases de données du marché (*MySQL*, *Miscrosoft SQL Server*, *Postgres*, etc.) bien qu'elle dispose d'une base de données intégrée (*SQLLite*). Il est possible de travailler avec les différents systèmes d'information géographique (*ArcGIS*, *PostGIS*, *Quantum GIS*, etc.) disponibles sur le marché pour afficher les informations entrées (modèles) ou le résultat des simulations. Ce logiciel est disponible en téléchargement libre sous licence publique générale GNU (GPL).

7.4.1.5 Un exemple de mise en œuvre d'*UrbanSim*

Dans l'étude de cas "la Tangentielle Nord", les rapports [246, 247, 248] relatifs au projet SIMAURIF (SIMulation de l'interAction Urbanisation-transport en Région d'Ile-de-France) indiquent que ce projet a exploité le modèle d'urbanisation *UrbanSim* avec succès mais non sans difficulté. En effet, la dimension historique et la densité importante de la ville de Paris sont différentes d'une ville américaine moyenne ayant des constructions beaucoup plus récentes et une organisation semble-t-il plus simple. D'autre part, les règles d'accès aux données sont différentes et parfois plus strictes et contraignantes en France qu'aux États-Unis. Enfin, ils ont été amenés au cours de ce projet à essayer plusieurs types de modèle afin d'en trouver un de qualité explicative suffisante compte tenu des données disponibles mais surtout de qualité prédictive et cette démarche s'est avérée très « gourmande » en temps de travail. Cette étude de cas permet également de mettre en évidence les limites de la modélisation : les modèles ne sont pas en mesure de prédire des événements exceptionnels, certains facteurs difficilement mesurables peuvent influencer les comportements et de nombreux facteurs humains sont impossibles à intégrer.

7.4.1.6 Conclusion

Malgré sa qualité incontestable, le modèle *UrbanSim* requiert une grande quantité de données en entrée qui ne sont pas toujours disponibles ou qui peuvent être soumises à des conditions strictes d'utilisation dans certains pays. D'autre part, il nécessite beaucoup d'efforts et de temps dans sa phase de calibrage pour être capable de reproduire l'évolution d'une ville au plus proche de la réalité entre l'année initiale et l'année de calage (cf. figure 7.10). Ensuite seulement, il peut être utilisé pour entrevoir un avenir hypothétique incapable de prendre en compte les événements exceptionnels (ce qui semble normal) ou de saisir toute la complexité inhérente aux comportements humains.

Pour compléter notre hypothèse de départ liée à l'utilisation éventuelle d'un modèle LUTI dans le cadre de cette thèse, nous pourrions imaginer différents paramétrages d'*UrbanSim* basés sur les résultats de différents calages, chacun étant lié à une morphologie de ville caractéristique. Pour une simulation, il faudrait alors se référer à un paramétrage particulier pour adopter un type de ville désiré. Néanmoins, les différents aspects évoqués relatifs à *UrbanSim* sont contraires à la philosophie même de la pré-programmation urbaine (voir section 4.1.2.1) caractérisée par un modèle simple pouvant être compris de tous avec un fonctionnement nécessitant un nombre limité de données en entrée. De plus, malgré le haut niveau reconnu à *UrbanSim*, il semble difficile de transposer un modèle américain aux villes européennes façonnées par de

[15] Open Platform for Urban Simulation

longues années d'histoire. Enfin et surtout, nous ne souhaitons pas simuler ou reproduire le comportement probable lié à l'évolution d'une ville, nous souhaitons plutôt optimiser son organisation globale à priori.

7.4.2 Le projet *GeOpenSim*

Intéressons-nous maintenant à une approche radicalement différente d'*UrbanSim* qui tente néanmoins de simuler l'évolution d'une ville en simplifiant le processus d'alimentation des modèles. Il s'agit d'une plateforme géographique de simulation qui s'inscrit dans un projet ANR[16] commencé en 2007 et qui s'est achevé en avril 2011. Le projet *GeOpenSim* [216] vise le développement d'une plateforme Open Source pour l'analyse et la simulation des dynamiques urbaines à l'aide de bases de données topographiques historiques. Ces dernières sont construites à l'aide de bases de données topographiques récentes ainsi que de photographies aériennes et/ou de cartes historiques. Ces bases de données sont analysées afin d'identifier :

- des règles d'évolution au niveau de la ville, du quartier ou de l'îlot ;
- des statistiques sur des objets spécifiques comme les bâtiments composant un îlot ou les îlots présents dans un quartier.

Chaque simulation peut ainsi être paramétrée à l'aide de règles d'évolution provenant de l'analyse automatique des données historiques, mais aussi de règles énoncées par des experts.

7.4.2.1 Approche

L'approche met en œuvre un système multi-agents qui comporte une hiérarchie d'agents topographiques (bâtiments, routes, cours d'eau, îlots, quartiers, ville) qui peuvent être construits, modifiés, fusionnés, découpés, restructurés et détruits au cours du temps. Le comportement de chaque agent est contrôlé par un ensemble de règles d'évolution, de contraintes et d'actions associées. *GeOpenSim* repose sur une structure hiérarchique (allant du bâtiment à l'agglomération) organisée par des réseaux de communication (les routes, les voies ferrées, etc.) et des graphes de voisinage (notamment entre bâtiments et entre les bâtiments et les routes) permettant une certaine autonomie à l'échelle micro (bâtiments, routes, etc.) et des comportements de régulation à différents niveaux macro (îlot, quartier, ville, etc.). Construit à partir d'un modèle existant, le système multi-agent hiérarchique a été adapté aux besoins de la simulation urbaine et plusieurs fonctionnalités ont été ajoutées, telles que les règles d'évolution et les méthodes de peuplement. Les règles d'évolution sont définies soit par analyse de données historiques, soit par des experts afin d'affecter des objectifs appropriés aux agents. Elles peuvent prendre en compte le voisinage des agents ainsi que leurs différents attributs (pour les îlots : la densité, le type d'îlot, la localisation de l'îlot dans la ville, etc.) Les méthodes de peuplement sont définies afin de modéliser les différentes stratégies que les agents peuvent utiliser pendant leur évolution (par exemple, la façon dont les îlots se densifient en reproduisant leur structure interne, en se restructurant ou en diversifiant leur structure). Le modèle *GeOpenSim* utilise des méthodes d'analyse spatiale permettant la construction automatique des objets composant les différents niveaux de la hiérarchie urbaine, ainsi que les relations entre ces objets. On peut ainsi reconstituer les objets ville, quartiers et îlots à partir des objets bâtiments et routes présents dans les bases de données topographies. D'autres structures telles que les alignements de bâtiments sont également automatiquement construites et utilisées dans la simulation.

Les objets géographiques sont évalués sur des critères morphologiques et c'est l'évolution de ces caractéristiques que l'on cherche à simuler :

- pour un bâtiment : sa surface, son élongation, sa concavité, son orientation, son type (habitation, industriel, etc.) ;
- pour une route : sa sinuosité, son type (rue, autoroute, etc.) ;
- pour un îlot : sa densité (quotient de sa surface bâtie et de sa surface totale), sa taille, son type (tissu urbain continu ou discontinu, industriel ou commercial, etc.), sa quantité d'espaces vides (par taille),

[16] Agence Nationale de la Recherche

indicateurs statistiques sur les bâtiments le composant, etc ;
- pour un quartier : sa surface, périphérique ou non, indicateurs statistiques sur les îlots le composant, etc ;
- pour une ville : sa surface totale, sa surface bâtie, la taille de ses réseaux de transport, la distribution de ses bâtiments par type, par taille, par forme, etc.

Le modèle *GeOpenSim* permet de simuler l'évolution de ces objets en les modélisant comme des agents, chaque agent ayant comme but de se transformer pour acquérir un état satisfaisant ses règles d'évolution. Pour cela, l'état cible d'un agent à une date $t + d_t$ est décrit sous forme de contraintes sur ses caractéristiques morphologiques ; les règles d'évolution sont utilisées uniquement pour calculer les valeurs buts de ces contraintes. L'objectif de chaque agent est de satisfaire ses contraintes afin d'acquérir un état à la date $t + d_t$ satisfaisant au mieux ses règles d'évolution. Lors de chaque activation, si l'état initial de l'agent n'est pas parfait, une liste d'actions pouvant potentiellement améliorer son état est calculée. Cette liste est fonction des contraintes insatisfaites de l'agent et de leur degré d'insatisfaction. Chaque action consiste à déclencher un algorithme de transformation géométrique sur l'agent afin de faire évoluer certaines de ses caractéristiques morphologiques. Ces essais sont effectués jusqu'à ce que l'agent atteigne un état parfait ou que toutes les actions possibles aient été essayées. Tous les agents ont le même cycle de vie leur permettant de satisfaire les contraintes relatives à leur niveau.

7.4.2.2 Conclusion

Cette plateforme opère à partir de données topographiques et utilise des règles d'évolution pour réaliser des simulations. Les résultats obtenus peuvent être comparés aux données réelles lorsque les simulations reposent sur des données historiques, permettant ainsi d'évaluer des hypothèses sur les dynamiques urbaines. L'utilisation de données topographies comme informations d'entrée du modèle semble particulièrement séduisante car ces informations sont largement répandues et de plus en plus précises. De plus, ces données peuvent être réactualisées régulièrement pour être réinjectées dans le modèle sans erreur de transcription grâce à des algorithmes capables de les interpréter automatiquement pour reconstituer tous les éléments urbains.

Faisant référence à notre hypothèse précédente, nous pourrions très probablement réorienter l'utilisation originale de cette plateforme pour générer une ville de toute pièce. Toutefois, le modèle *GeOpenSim* s'attache à une représentation géométrique de la ville et travaille à un niveau très fin allant jusqu'au bâtiment. Dans cette thèse, nous devons au contraire nous focaliser en priorité sur les caractéristiques fonctionnelles de la ville dont le niveau de granularité le plus fin correspond à l'îlot urbain. Et tout comme *UrbanSim*, *GeOpenSim* s'attache essentiellement à deviner une évolution probable d'un espace urbain sans nécessairement chercher à l'optimiser.

7.4.3 CommunityViz

CommunityViz [284] est un logiciel de planification urbaine proposé par la société *placeways*[17] qui se présente sous la forme d'un module d'extension (plugin) pouvant s'insérer et étendre les capacités du système d'information géographique « *ArcGIS for Desktop* » de la société *Esri*[18]. Ce logiciel permet d'accompagner les acteurs de l'urbain (urbanistes, gestionnaires des ressources naturelles, administrations locales, organismes d'aménagement métropolitain, entreprises privées, ...) en les aidant à simuler le futur d'une ville ou d'une région. Il permet de créer des plans ou des propositions de développement tout en présentant sur la zone d'étude les impacts économiques, environnementaux et sociaux. A partir de ces impacts, et en comparant différentes alternatives, la population est mieux informée et les décisionnaires peuvent prendre des décisions éclairées en ayant connaissance des conséquences. *CommunityViz* dispose d'une large gamme de

[17] http://placeways.com/communityviz/index.html.
[18] Environmental Systems Research Institute : http://www.esri.com/

fonctionnalités pour aider les décisions de localisation, de type et de forme que peut prendre un développement urbain.

Le logiciel s'adresse à différentes catégories d'acteurs et chacune selon son niveau peut tirer parti de l'outil. Les membres du public sans connaissance particulière en urbanisme peuvent profiter des visualisations interactives lors des réunions publiques, ces présentations étant également accessibles sur un site web dédié. *CommunityViz* propose des outils faciles à utiliser et accessibles au plus grand nombre pour produire des analyses de base et des visualisations. Concernant les experts, comme les urbanistes et les professionnels en systèmes d'information géographique, ils peuvent utiliser le logiciel pour créer des études qui seront ensuite exploitées par les entreprises privées, les administrations et les audiences publiques. Cette suite logicielle est composée de deux modules principaux : (i) *Scenario 360* qui fournit les outils interactifs d'analyse et le framework d'aide à la décision et (ii) *Scenario 3D* qui offre un moyen de créer une information riche au travers de scènes 3D interactives.

7.4.3.1 Scenario 360

Scenario 360 constitue la pièce centrale de *CommunityViz*. Elle a été conçue selon les principes suivants :

- créer et envisager des hypothèses selon la formule : « que se passerait-il si ... ? »,
- évaluer graphiquement les impacts économiques, environnementaux et sociaux des aménagements proposés (plan d'occupation des sols, investissements dans des infrastructures, ...),
- faire des hypothèses pouvant être rapidement remises en cause,
- travailler à l'échelle d'un site, d'un quartier, d'une ville ou d'une région,
- prendre des décisions globales et éclairées,
- se connecter à des outils de visualisation 3D.

Scenario 360 regroupe les fonctionnalités suivantes, en considérant que certains composants travaillent ensemble (i.e. l'entrée d'un modèle correspond à la sortie d'un autre) :

- « *Land Use Designer* » : permet de dessiner sur une carte les types d'utilisation du sol (parmi une liste disponible en standard ou spécifique) et de voir instantanément les impacts correspondants (sociaux, économiques et environnementaux). Les différents scénarios construits permettent de donner un sentiment à l'audience sur comment chaque alternative peut affecter le futur de leur communauté ;
- « *Build-Out Wizard* » : calcule la capacité de développement d'un territoire. Les analyses correspondantes montrent combien d'immeubles de quel type peuvent être construits sur chaque emplacement en fonction d'un règlement communautaire donné sur l'utilisation des sols, la densité, le renouvellement urbain, les hauteurs, etc. Une fonction d'allocation permet de projeter la localisation du développement au fil du temps ;
- « *Suitability Wizard* » : permet de calculer automatiquement un score lié à chaque emplacement pour la localisation des activités (habitation, activité commerciale, activité artisanale, ...) selon différents facteurs de qualité pouvant être pondérés (facteur d'attraction ou de répulsion par rapport à la proximité d'une route, d'une ligne d'égoût, d'un cours d'eau, une pente, ...) ;
- « *TimeScope* » : permet de voir année après année l'évolution des changements. Calcule où et quand un nouveau développement apparaîtra en fonction des hypothèses fournies concernant les taux et les modèles de croissance (par exemple : 3 % de croissance par an, développement prioritaire des espaces libres proches des routes, ...) ;
- *des outils pour visualiser de multiples scénarios* côte à côte sur une analyse ;
- *des outils pour créer facilement des illustrations sur le web* à partir d'une analyse pour une représentation interactive en 2D ou en 3D via *Google Earth* ;
- *des graphiques dynamiques, alertes et autres affichages* actualisés en temps réel en fonction des alternatives et des actions.

Plus globalement, ce module recherche l'efficacité à travers un processus de planification communautaire axé sur une approche visuelle et collaborative prenant place lors des séances publiques. Il fonctionne

comme une extension d'*ArcMap*[19] et s'intègre par conséquent dans la suite « *ArcGIS for Desktop* ». *CommunityViz* peut donc exploiter toutes les données géographiques et les plans accessibles ainsi que toutes les fonctionnalités disponibles de cette suite logicielle.

7.4.3.2 Scenario 3D

Le module *Scenario 3D* vient accompagner *Scenario 360* dans la suite *CommunityViz*. *Scenario 3D* permet de créer des scènes 3D réalistes, interactives et partageables. Il suffit pour cela de spécifier une carte *ArcGIS* et de dire comment les fonctionnalités 2D doivent être représentées dans la scène 3D. Lorsqu'on visualise une scène, on peut alors se déplacer comme si on s'y trouvait, en marchant, en volant et en regardant autour de soi. On peut également sélectionner un objet en cliquant dessus pour lire des informations complémentaires où les entendre parler. Les constructions, arbres et routes apparaissent dans les détails de façon photo-réaliste, en suivant le relief du terrain. Les ombres, lumières et effets de brouillard viennent renforcer le côté réaliste des lieux : l'idée étant de traduire la sensation qui se dégage de l'endroit que l'on perçoit.

Scenario 3D est composé de deux parties : un outil d'export et un outil de visualisation 3D. Le module d'export est une extension d'*ArcGIS* tandis que l'outil de visualisation est une application indépendante permettant de visualiser librement les scènes 3D générées. Par rapport à un export *Google Earth* disponible dans *Scenario 360*, le niveau de détail fourni par *Scenario 3D* est supérieur, tout comme les possibilités de navigation au sol, les effets disponibles (ombre, lumière, brouillard) ou le nombre de formats 3D supportés. En contrepartie, le temps de génération nécessaire est plus long et il est préférable de circonscrire la zone d'étude au quartier ou au site.

7.4.3.3 Conclusion

CommunityViz intègre une large gamme d'outils disponibles pour les urbanistes, les décisionnaires ou le grand public, y compris des outils de conception automatisés et interactifs.

Néanmoins, pour instruire un projet, cette suite logicielle nécessite l'introduction de données initiales conséquentes. Cela peut concerner la population existante, les polygones délimitant les parcelles, l'utilisation des sols, les réseaux de transport, l'environnement naturel, les canalisations (eau, égouts, ...), la fiscalité, l'économie locale, l'ensemble de l'environnement bâti existant, etc. Même si l'intégralité de ces informations n'est pas obligatoire, ceci représente un travail conséquent et la mobilisation de compétences spécifiques avant même de pouvoir faire la moindre simulation. D'autre part, cet outil travaille à un niveau de détail très fin (habituellement la parcelle) et l'aspect géométrique y reste très présent.

Enfin, certaines fonctionnalités automatiques proposées dans *CommunityViz* reposent sur des techniques d'optimisation combinatoire (par exemple : pour identifier une combinaison de sites naturels à traiter qui apportent un maximum de bénéfice par rapport à un budget donné ou d'autres contraintes). Toutefois, ces techniques sont peu documentées et elles deviennent vite inopérantes dès que le problème atteint une taille trop importante (i.e. au delà de 500 éléments à prendre en compte).

7.5 Les problèmes de positionnement

Indépendamment des modèles urbains spécialisés que nous avons abordés jusqu'à présent, notre problématique s'apparente plus spécifiquement à un problème de positionnement dans le sens où l'on se pose la question suivante : « *Où allons nous positionner les choses ?* » [80, 218].

Les études relatives à la théorie de la localisation remontent à 1909 avec les travaux d'Alfred WEBER qui cherchait à positionner un unique entrepôt tout en minimisant la distance totale entre son lieu d'implantation et différents clients. Par la suite, cette théorie s'est développée sans sursaut au gré de quelques

[19] Module de la suite « *ArcGIS for Desktop* » capable de représenter les informations géographiques sous forme d'un ensemble de couches (layers) et les autres informations sous la forme de cartes.

applications. Il faudra attendre 1964 pour que le sujet suscite à nouveau un regain d'intérêts avec la publication de S.L. HAKIMI [124] qui s'intéressait au positionnement de centres de commutation dans un réseau de communications et de postes de police sur un réseau routier.

Les problèmes de localisation peuvent être décrits à partir de quatre composants :

- les clients qui sont supposés être déjà positionnés à des points ou sur des routes ;
- les équipements que l'on cherche à positionner ;
- un espace où sont localisés les clients et les équipements ;
- une métrique qui indique les distances ou le temps entre les clients et les équipements.

Ce sujet correspond à des problèmes d'optimisation très étudiés en recherche opérationnelle compte tenu des enjeux théoriques et applicatifs qu'ils suscitent et des répercussions que ce type de décision engendrent [156]. En effet, des choix de positionnement sont fréquemment réalisés à tous les niveaux de l'organisation humaine, aussi bien au niveau de l'individu qu'au niveau des organisations internationales en passant par les ménages ou les entreprises. Ils impliquent souvent des ressources conséquentes avec des impacts économiques et environnementaux importants sur le long terme (pollution, congestion du trafic, etc.). D'un point de vue scientifique, ces problèmes sont souvent extrêmement difficiles à résoudre, sachant que même les modèles les plus classiques sont $\mathcal{NP}$-*complet* et difficilement surmontables pour de grandes instances. Il a fallu attendre l'avènement d'ordinateurs suffisamment puissants pour susciter l'intérêt de formalisation et d'implémentation de ces modèles de positionnement compte tenu de leur complexité de calcul. Finalement, ces modèles sont spécifiques à l'application considérée, ce qui rend leur structure (objectifs, contraintes et variables) particulière au problème étudié. Les décisions relatives aux problèmes de positionnement font régulièrement intervenir de nombreuses parties prenantes et doivent considérer de multiples objectifs souvent conflictuels [60].

Le lecteur intéressé pourra se référer au chapitre « Discrete Network Location Models » de J. *Current* et al. [60] pour obtenir une description détaillée des modèles classiques de positionnement connus sous les noms : *set covering*, *maximum covering*, *p-center*, *p-dispersion*, *p-median*, *fixed charge*, *hub* et *maxisum*.

Le livre « *Facility Location* » [90] constitue également une excellente source d'informations sur les nombreux problèmes de positionnement qu'il aborde.

7.5.1 Cadre général d'un problème de positionnement

Dans le cadre du problème de positionnement d'équipements (Facility Location Problem, FLP), on cherche à placer un certain nombre d'*équipements* par rapport à des *demandes* de façon à optimiser un ou plusieurs *critères*, tels que la distance, la satisfaction de la demande ou encore les temps de parcours [199]. Dans ce contexte, le terme *équipement* est à prendre au sens large et peut faire référence à des entrepôts, des hôpitaux, des usines, des écoles, etc.

Ce type de problème soulève trois questions principales :

- quels sont les équipements à positionner : nombre et nature ;
- quels sont les emplacements disponibles : nombre et forme ;
- quels sont les critères à prendre en compte pour positionner chaque équipement sur un emplacement.

Dans la famille des problèmes de positionnement, le problème d'affectation quadratique est un problème très réputé à la fois pour ses nombreuses applications pratiques et parce qu'il est particulièrement difficile à résoudre. Nous proposons d'étudier plus spécifiquement ce problème compte tenu de l'intérêt qu'il suscite et de sa proximité avec notre problématique.

7.5.2 Problème d'affectation quadratique (Quadratic Assignation Problem, QAP)

Le problème d'affectation quadratique est un problème d'optimisation combinatoire introduit pour la première fois par C. KOOPMANS et M. BECKMANN [155] en 1957. Il s'agit d'un problème $\mathcal{NP}$-*difficile* qui n'est pas ρ-approximable quelle que soit la constante ρ choisie [238].

FIGURE 7.11 – Affectation (assignment) $\pi = (3, 2, 4, 1)$. D'après R.Z. FARAHANI et al. [90].

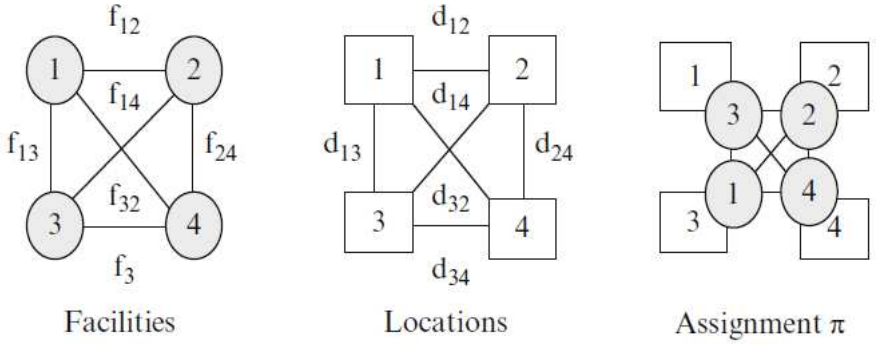
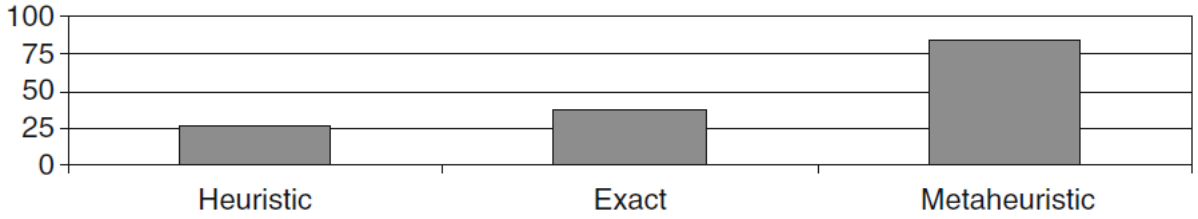


FIGURE 7.12 – Publications : proportions des solutions techniques proposées pour les QAP. D'après R.Z. FARAHANI et al. [90].



7.5.2.1 Modélisation du problème

Ce problème peut être défini ainsi [187] (cf. figure 7.11) : un ensemble de n équipements ("*facilities*") communiquant entre eux doivent être positionnés à moindre coût sur n emplacements prédéterminés ("*locations*"). La distance entre les emplacements k et l est notée d_{kl} et le flux échangé entre les équipements i et j est noté f_{ij}. Le coût d'interaction entre l'équipement i placé sur le site k et l'équipement j placé sur le site l est considéré comme proportionnel au produit du flux échangé entre les deux équipements (f_{ij}) par la distance entre les deux emplacements (d_{kl}). Le problème consiste à affecter ("*assignment*") à chaque emplacement un seul équipement tout en minimisant le coût :

$$C = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}\, d_{p(i)p(j)} \tag{7.1}$$

où $p(i)$ représente l'emplacement assigné à l'équipement i.

Pour des questions de simplification, cette modélisation ne prend pas en compte le coût d'implantation d'un équipement sur un emplacement donné.

7.5.2.2 Résolution par des méthodes de RL

La difficulté de ce problème s'illustre par la faible taille des instances pouvant être résolues de façon exacte. En effet, au delà d'une taille située aux alentours de 30 équipements, l'explosion combinatoire est telle qu'une résolution exacte devient illusoire. Face à des instances de taille élevée, il est donc nécessaire de recourir à une méthode de résolution approchée [187, 53, 90].

La nécessité de disposer de méthodes approchées performantes a incité de nombreux chercheurs à développer des solutions métaheuristiques pour s'attaquer à ce problème (voir figure 7.12). En dehors des algorithmes évolutifs, les méthodes de recherche locale sont celles qui apparaissent dans les travaux les plus représentatifs.

TABLE 7.1 – Pourcentages d'erreur des meilleures métaheuristiques appliquées au QAP. RobuTS (Recherche tabou robuste [263]), ReacTS (Recherche tabou réactive [17]), Rec-Sim (Recuit simulé [54]), Hybrid-Gen (Algorithme génétique [94]), Hybrid-Ant (Méthode de colonies de fourmis [100]). D'après T. MAUTOR [187].

Intervalle d'erreur	**RobuTS**	**ReacTS** (avec 22 instances)	**Rec-Sim**	**Hybrid-Gen**	**Hybrid-Ant**
<0.01%	12	-	1	13	13
0.01% - 0.1%	9	1	4	7	7
0.1% - 1%	12	17	18	15	10
1% - 10%	5	4	13	3	8
≥10%	-	-	2	-	-

Mais avant même d'aborder les résultats obtenus par les meilleures méthodes de recherche locale pour ce problème, il est primordial d'identifier une structure de voisinage adaptée au problème à traiter.

Structure du voisinage Les méthodes de recherche locale développées pour le problème d'affectation quadratique utilisent très majoritairement une structure de voisinage très simple, basée sur les 2-échanges (ou transpositions), c'est à dire toutes les solutions pour lesquelles la seule différence avec la solution courante consiste à échanger le placement de deux équipements [187]. Selon Thierry MAUTOR, deux raisons principales justifient le choix de cette structure de voisinage. D'une part, le voisinage est de taille raisonnable : composé de $\frac{n(n-1)}{2}$ solutions. D'autre part, la variation du coût occasionnée par un 2-échanges peut être calculée très rapidement grâce à un calcul incrémental.

Notons :

- p : la solution courante ;
- p_{uv} : la solution obtenue à partir de p en échangeant le placement des équipements u et v ;
- $\triangle_p(u,v)$: la variation de coût occasionnée par l'échange du placement des équipements u et v, on a $\triangle_p(u,v) = \text{Cout}(p_{uv}) - \text{Cout}(p)$,

La variation de coût occasionnée par l'échange de placement des équipements u et v peut être calculée selon la formule (en considérant des matrices D et F symétriques) :

$$\triangle_p(u,v) = 2 \sum_{j=1, j\neq u, , j\neq v}^{n} (f_{uj} - f_{vj})(d_{p(v)p(j)} - d_{p(u)p(j)}) \tag{7.2}$$

De plus, lorsqu'un mouvement a été effectué en échangeant le placement des équipements x et y, différents de u et v, la variation du coût occasionnée par l'échange de placement des équipements u et v devient :

$$\triangle_{p_{xy}}(u,v) = \triangle_p(u,v) + 2(f_{ux} + f_{vy} - f_{uy} - f_{vx})(d_{p(v)p(y)} + d_{p(u)p(x)} - d_{p(u)p(y)} - d_{p(v)p(x)}) \tag{7.3}$$

Avec un voisinage complexe, le coût pour chaque solution voisine devrait être évalué en $O(n^2)$. Avec un voisinage en 2-échanges, lors d'une première itération, toutes les solutions voisines de la solution initiale sont évaluées en $O(n)$ (cf. l'équation 7.2). Ensuite, une fois cette initialisation réalisée, toutes les itérations suivantes s'effectuent en $O(1)$ en mettant à jour les $\triangle_p(u,v)$ calculés à l'itération précédente (cf. l'équation 7.3). L'évaluation complète d'un voisinage se fait alors en $O(n^2)$.

Résultats liés aux méthodes de RL Parmi les métaheuristiques existantes, les méthodes de recuit simulé et de recherche tabou ont été appliquées dès leurs origines aux problèmes d'affectation quadratique. Dès le

début, les résultats remportés à partir de ces deux méthodes ont très largement surclassé ceux obtenus par les méthodes utilisées jusque là et basées sur des méthodes gloutonnes ou des procédures de descente. Sur des problèmes de petite taille (inférieure à 15 équipements), ces nouvelles approches étaient très souvent capables de trouver la solution optimale [187]. Avec des problèmes de taille plus élevée (inférieure à 50 équipements), il a été confirmé depuis, grâce aux progrès réalisés par les méthodes complètes, que les résultats obtenus étaient capables de converger la plupart du temps vers la valeur optimale supposée. Pour des problèmes de plus grande taille (comprise entre 50 et 150 équipements), les solutions optimales restent encore inconnues et les approches métaheuristiques continuent de se mesurer entre elles pour améliorer les meilleurs résultats connus.

Malgré leur ancienneté, les méthodes tabou proposées par TAILLARD [263] (recherche tabou robuste, 1991) ou par BATTITI et TECCHIOLLI [17] (recherche tabou réactive, 1994) sont très robustes et restent des références difficiles à surclasser pour le problème d'affectation quadratique. Néanmoins, si on compare entre elles les meilleures métaheuristiques appliquées à QAP (recuit simulé, recherche tabou, GRASP, méthodes de colonies de fourmis, ...), on constate qu'elles obtiennent des résultats presque similaires et très proches des meilleures solutions connues (cf. table 7.1).

7.6 Conclusion

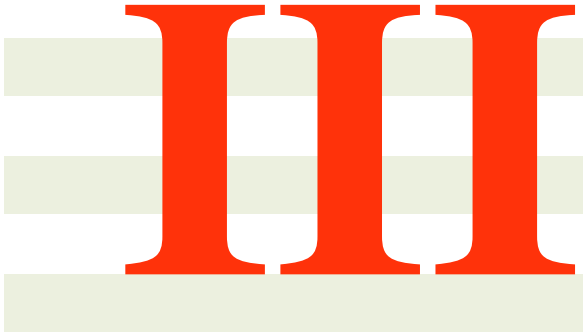
« *Le monde de la recherche en urbanisme et en aménagement s'attelle depuis plusieurs années à une tâche complexe, voire à une gageure, fondée sur des savoirs interdisciplinaires à mobiliser et à intégrer dans l'outillage technique des sciences humaines : la modélisation de la ville* » [7].

En effet, cette tâche nécessite la mobilisation d'équipes pluridisciplinaires importantes qui doivent travailler en collaboration étroite compte tenu du large spectre de connaissances et d'expertises requis pour aboutir à un résultat opérationnel.

Les projets actuels liés aux problématiques de la modélisation urbaine s'intéressent plus particulièrement aux évolutions permanentes qui s'opèrent dans les villes au niveau du temps et de l'espace. On parle ainsi de modèle "dynamique". Cette approche très répandue nécessite une quantité importante de données historiques en amont pour alimenter les modèles ainsi que la mise en place de règles énoncées par des experts du domaine. Malgré la pertinence de ces modèles et les efforts consentis, nous devons néanmoins rester vigilants par rapport à cette ambition car de nombreux projets architecturaux ou liés à des politiques de la ville et du transport se sont soldés par un échec [7].

D'après le philosophe Maurice BLONDEL, « *L'avenir ne se prévoit pas, il se prépare* ». Or, pour Gaston BERGER, « *L'avenir n'est pas seulement ce qui peut arriver ou ce qui a le plus de chances de se produire. Il est aussi, dans une proportion qui ne cesse de croître, ce que nous aurons voulu qu'il fût* » [122].

Bien que notre problématique se situe à un stade précoce du processus de modélisation des villes, notre approche s'inscrit clairement dans une démarche volontariste. C'est pour cela que nous proposons, dans nos contributions, un modèle urbain durable basé sur des contraintes, composées essentiellement de préférences qui garantiront un équilibre entre les enjeux sociaux, environnementaux et économiques. Ensuite, nous proposons de résoudre le problème en cherchant parmi les solutions admissibles, la meilleure solution possible. Dans ce sens, notre façon d'aborder le sujet par l'optimisation vient compléter les approches classiques qui cherchent à prévoir ce qui a le plus de chances de se produire : notre objectif étant plutôt de trouver le meilleur avenir possible. Et comme la modélisation n'est qu'une représentation simplifiée du monde, nous permettons aux décideurs d'intervenir sur le résultat des simulations pour améliorer ce qui peut l'être et prendre en compte les impératifs du moment ou les notions difficilement modélisables. Une rétroaction entre l'homme et la machine peut alors se mettre en marche.



Contributions

8

Modèle

8.1 Introduction

La pré-programmation permet aux décideurs et aux parties prenantes (aménageurs, urbanistes, habitants, ...) de se mettre d'accord sur les grandes lignes d'un projet urbain préliminaire avant d'engager des études importantes, longues et coûteuses. Pour modéliser notre problème, nous allons commencer par décrire le modèle urbain de pré-programmation durable tel qu'il a été spécifié par les urbanistes et les architectes au fil de nos échanges. Dans cette première étape, l'enjeu consiste à relever et à expliquer les notions fondamentales que nous avons décidé d'exploiter. Face à la complexité de l'organisation d'un espace urbain, un travail conséquent a dû être réalisé en collaboration avec les spécialistes de la ville pour comprendre cette complexité et en extraire les éléments significatifs.

Dans une seconde étape, nous décrivons le modèle de contrainte pour notre problème d'aménagement urbain. Il s'agit d'un problème particulier de positionnement de formes urbaines sur un territoire pour constituer une ville cohérente. Bien que l'on puisse faire des analogies avec des problèmes de positionnement classiques comme le problème d'affectation quadratique, notre problème est constitué de nombreuses contraintes spécifiques liées à l'aménagement urbain.

Ces différentes contraintes sont essentielles car elles permettent de décrire les règles qui régissent une pré-programmation urbaine. Elles participent individuellement ou collectivement à la lutte contre l'étalement urbain, la viabilité économique, la mixité fonctionnelle, le bien être des habitants, etc. Compte tenu de leur importance, nous proposons pour chaque contrainte une description générale et une description formelle accompagnées d'une représentation schématique pour illustrer chaque situation.

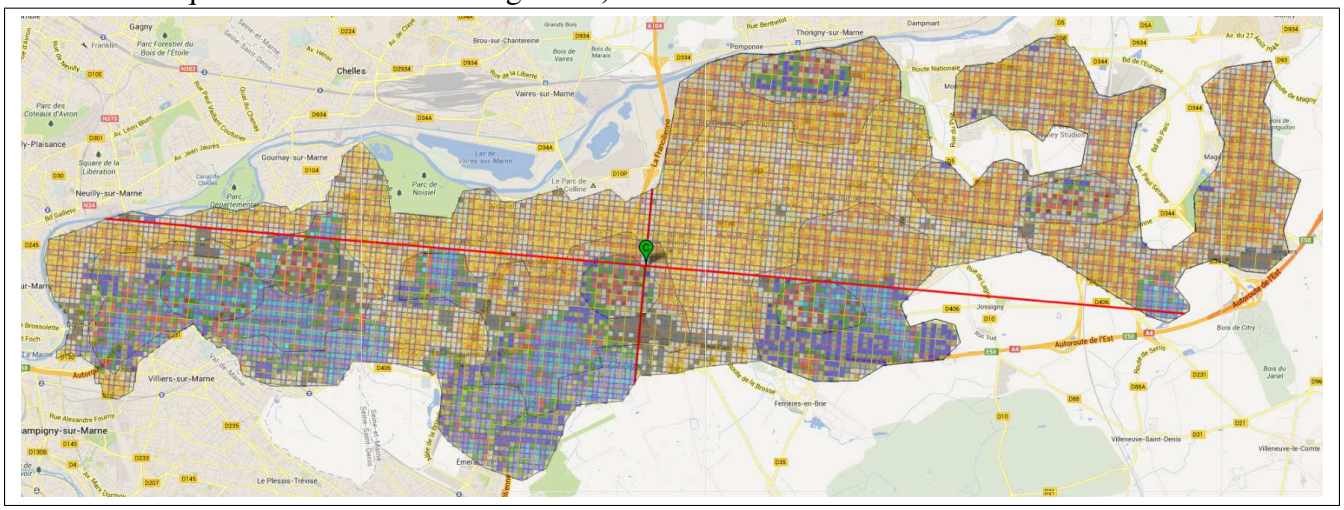
8.2 Modèle urbain de pré-programmation durable

La pré-programmation urbaine consiste d'abord à sélectionner le nombre et la nature de toutes les formes urbaines[1] composant la ville puis à les organiser spatialement, en prenant en compte les aspects environnementaux, sociaux et économiques propres au développement durable (étalement urbain, mixité fonctionnelle, mobilité, énergies, ...).

Plusieurs paramètres importants doivent être contrôlés : la densité de la population, les contraintes liées au territoire à aménager et le taux d'emploi. Le taux d'emploi est calculé en divisant le nombre d'emplois

[1] La forme urbaine correspond à l'utilisation qui est faite de l'espace ou du bâti sur chaque îlot urbain (habitat, commerce, ...), à ne pas confondre à la forme que peut prendre une ville entière (par exemple, forme radioconcentrique).

FIGURE 8.1 – Formes urbaines réparties automatiquement sur la zone expérimentale de la ville de Marne-la-Vallée, en prenant en compte les contraintes d'un développement urbain durable exprimées par les urbanistes. D'après les travaux du projet SUSTAINS - schéma fourni par le LIMSI (Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur).



par le nombre de personnes en âge de travailler et vivant dans la zone d'étude. Un taux d'emploi proche de "1" correspond à une situation idéale où chaque résident peut accéder à un travail tout en vivant dans la zone, minimisant ainsi les déplacements. Un taux d'emploi éloigné de "1" a des conséquences graves, induisant un trafic important pour entrer et sortir de la ville.

8.2.1 Modèle urbain

En urbanisme, « *la modélisation consiste à simplifier la réalité du monde pour mieux comprendre comment les décisions et les événements interagissent les uns avec les autres. Elle permet alors de tester des solutions susceptibles d'influencer ou d'orienter les décisions politiques et les stratégies pouvant conduire à un futur souhaitable* » [8]. Dans cette thèse, le modèle urbain proposé par les urbanistes repose sur les notions essentielles représentées par les *îlots*, les *centralités*, les *formes urbaines* et les *intensités*, et il intègre, dans une démarche systémique, les contraintes urbaines majeures liées à un développement urbain durable.

8.2.1.1 Îlot

L'îlot urbain dont les contours sont définis par les routes est le niveau de granularité le plus fin sélectionné par les urbanistes. Sa dimension est fixée de façon à ce qu'un habitant puisse facilement en faire le tour à pied. Les îlots sont représentés dans un quadrillage orthogonal avec une dimension de 80 mètres de côté. Il est supposé que chaque îlot est associé à une seule forme urbaine (habitat, industrie, commerce, école, etc).

8.2.1.2 Centralité

Une notion très importante pour les urbanistes concerne la centralité : une place structurante (ou lieu central) composée de quatre îlots tout au plus qui donne son nom au quartier. La centralité apparaît comme une propriété fondamentale qui participe à la formation des agglomérations urbaines. La valorisation du capital économique, social et symbolique accumulé localement suscite des investissements visant à renforcer l'accessibilité du lieu central, au fur et à mesure de sa croissance, par rapport aux lieux avec lesquels il est en concurrence. Et ce surcroît d'accessibilité rend le lieu encore plus attractif pour de nouvelles activités. Le

sentiment d'appartenir à un quartier coïncide avec la zone d'influence du lieu central, estimée à 300 mètres de rayon.

8.2.1.3 Forme urbaine

Dans notre contexte, la forme urbaine correspond à un type d'utilisation dominant qui est fait de l'espace ou du bâti pour un îlot complet. Néanmoins, une forme urbaine dite dominante peut tout à fait intégrer dans sa déclaration une certaine mixité fonctionnelle (proportion d'habitat, de commerce, etc.). Vingt huit formes urbaines ont été retenues (correspondant à quatre groupes : habitat, activité économique, infrastructure et élément fixe) :

- maison individuelle ;
- maison de ville ;
- habitat intermédiaire ;
- habitat collectif (R+4)[2] ;
- habitat collectif (R+7)[2] ;
- bureau, petit immeuble (R+4)[2] ;
- bureau, grand immeuble (R+7)[2] ;
- parc d'entreprises ;
- industrie ;
- commerce et service de centre ville ;
- hypermarché
- école maternelle et élémentaire ;
- collège ;
- lycée ;
- équipement administratif ;
- équipement technique ;
- équipement sportif ;
- espace vert ;
- espace de respiration ;
- rivière ;
- route résidentielle ;
- route intermédiaire ;
- avenue ;
- terrain accidenté ;
- espace naturel ;
- voie préexistante ;
- voie de chemin de fer ;
- zone inconstructible ;

8.2.1.4 Intensité

L'intensité représente une échelle de mesure liée au niveau de densité de la population sur une zone donnée. Elle permet également de traduire le niveau d'activité ou d'animation d'une région. Une intensité urbaine élevée correspond à un quartier vivant, dense, mixte et pour lequel la marche à pied est le moyen de déplacement le plus simple pour accéder à toutes les fonctions urbaines essentielles. On retrouve ce niveau d'intensité urbaine au centre ville tandis qu'une intensité urbaine faible est plutôt liée à un quartier à dominante pavillonnaire. Six niveaux d'intensité sont définis allant de « 1 » pour l'intensité la plus faible à « 6 » pour la plus élevée.

8.3 Modèle de contrainte pour le problème d'aménagement urbain

Pour un territoire donné, le modèle urbain nous fournit le nombre de formes urbaines de chaque type, pour chaque niveau d'intensité. L'objectif consiste alors à répartir spatialement les formes urbaines sur l'ensemble du territoire. Nous identifions ce problème comme un *problème d'aménagement urbain*.

Notre première tâche consiste à fournir une formalisation des contraintes et des préférences relatives aux propriétés d'une ville durable. Les urbanistes expriment naturellement les interactions entre différentes formes urbaines comme des préférences de positionnement au lieu de contraintes dures (par exemple, les industries souhaitent être à proximité d'une voie ferrée ou d'un axe routier, mais elles préfèrent rester éloignées des écoles ou des habitations). Nous exprimons donc ces préférences dans des fonctions de coût et représentons le problème comme un problème d'optimisation combinatoire. Mais il y a également des contraintes dures qui restreignent strictement les positions de certaines formes urbaines (Par exemple, on ne

[2] (R+"x") signifie : rez-de-chaussée + "x" étages.

peut pas placer une maison individuelle, associée à une intensité urbaine très faible, en plein centre urbain lié à une intensité urbaine très forte).

En fin de compte, notre problème englobe un grand nombre de contraintes, certaines étant très spécifiques. Nous distinguons dans notre présentation les plus importantes, les *contraintes générales* qui s'appliquent à toutes les variables, et les *contraintes spécifiques* qui s'appliquent uniquement à certaines zones ou à certaines formes urbaines. Dans la suite, nous détaillons l'ensemble des fonctions de coûts et des contraintes retenues pour modéliser notre problème d'aménagement urbain.

8.3.1 Représentation de la ville par une grille

Nous représentons la ville par une grille régulière où chaque cellule correspond à un îlot urbain. L'objectif consiste à affecter à chaque cellule de la grille une forme urbaine parmi les formes possibles.

Certaines cellules spécifiques sont associées à des éléments fixes (routes, zones non constructibles, etc.). Elles sont donc considérées comme non libres et ne sont pas prises en considération. Pourtant, elles sont conservées sur la carte car elles peuvent interférer avec les autres formes urbaines. De plus, chaque cellule est affectée à un niveau d'intensité spécifié manuellement par les urbanistes (au tout début du processus de conception). Ceci permet par exemple de représenter les centralités sur la carte (zones commerciales, places animées, etc.).

Par souci de simplicité, la grille est vue comme une matrice rectangulaire de cellules de taille $(x \times y)$, mais cette représentation est plus symbolique que géométrique, dans la mesure où c'est le voisinage qui est important et le placement relatif des formes urbaines, pas la géométrie rectangulaire.

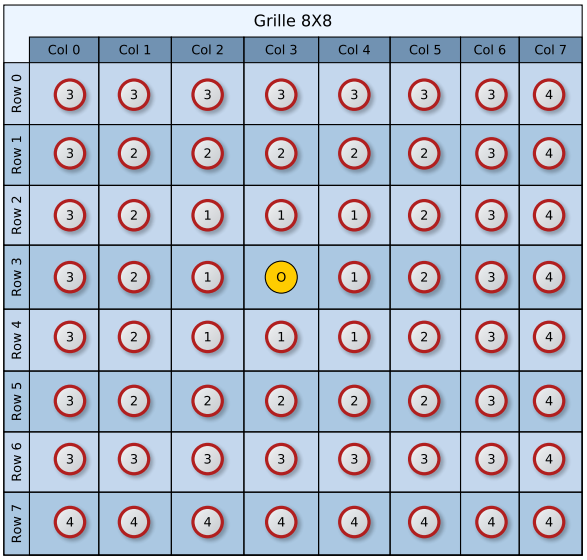
Chaque cellule correspond à une variable dont la valeur est sélectionnée parmi toutes les formes urbaines. Nous notons $V_{l,c}$ la variable correspondant à la cellule en ligne l, colonne c. Les formes urbaines sont encodées avec les valeurs entières suivantes :

0. non défini ;
1. maison individuelle ;
2. maison de ville ;
3. habitat intermédiaire ;
4. habitat collectif (R+4) ;
5. habitat collectif (R+7) ;
6. bureau, petit immeuble (R+4) ;
7. bureau, grand immeuble (R+7) ;
8. parc d'entreprises ;
9. industrie ;
10. commerce et service de centre ville ;
11. hypermarché
12. école maternelle et élémentaire ;
13. collège ;
14. lycée ;
15. équipement administratif ;
16. équipement technique ;
17. équipement sportif ;
18. espace vert ;
19. espace de respiration ;
20. rivière ;
21. route résidentielle ;
22. route intermédiaire ;
23. avenue ;
24. terrain accidenté ;
25. espace naturel ;
26. voie préexistante ;
27. voie de chemin de fer ;
28. zone inconstructible ;

Dans notre modèle, on s'intéresse à un problème d'optimisation sous contraintes où certaines contraintes sont traduites en fonctions de coût alors que d'autres restreignent les affectations possibles lors de la résolution. Lorsqu'une forme urbaine est assignée à une cellule, un coût pour cette cellule peut être calculé à partir de chaque contrainte applicable. La somme de ces coûts permet d'obtenir un coût total pour la cellule en question. L'objectif consiste à minimiser la somme des coûts sur l'ensemble des cellules, le coût global étant noté $Cost$. On cherche alors à déterminer pour chaque cellule la forme urbaine $V_{l,c}$ qui permet de minimiser le coût global $Cost$ issu du coût de chaque cellule $V_{l,c}$ pour chaque contrainte ct applicable :

$$Cost = \sum_{l=0}^{x-1} \sum_{c=0}^{y-1} \sum_{ct=1}^{z} \left(w_{ct} * Cost_{ct} \left(V_{l,c} \right) \right) \tag{8.1}$$

FIGURE 8.2 – Illustration de la distance de *Tchebychev* à partir de la cellule marquée par un « O » et positionnée en ligne 3, colonne 3 : chaque colonne est repérée par sa distance (allant de 1 à 4) par rapport à la cellule « O ».



où : z correspond au nombre de contraintes définies et w_{ct} au poids associé à chaque contrainte ct.

Par rapport à notre modèle, abordons dès à présent l'ensemble des notations générales, en lien avec la représentation en grille, qui seront reprises par la suite dans la définition des contraintes.

8.3.1.1 Distance de *Tchebychev*

Pour certaines contraintes, nous utilisons la distance de *Tchebychev* (cf. figure 8.2) qui s'exprime en nombre de cellules. Soit deux cellules $V_{i,j}$ et $V_{k,l}$, la distance de *Tchebychev* séparant les cellules $V_{i,j}$ et $V_{k,l}$ est :

$$\mathcal{D}_{\text{tch}}(V_{i,j}, V_{k,l}) = \max\left(|k-i|, |l-j|\right) \tag{8.2}$$

8.3.1.2 Distance de *Manhattan*

Pour d'autres contraintes, nous utilisons la distance de *Manhattan* (cf. figure 8.3) correspondant à la distance parcourue en empruntant les rues agencées selon un quadrillage (en référence à la structure des villes américaines, typiquement le quartier de Manhattan à New York). Soit deux cellules $V_{i,j}$ et $V_{k,l}$, la distance de *Manhattan* séparant les cellules $V_{i,j}$ et $V_{k,l}$ est :

$$\mathcal{D}_{\text{man}}(V_{i,j}, V_{k,l}) = |k-i| + |l-j| \tag{8.3}$$

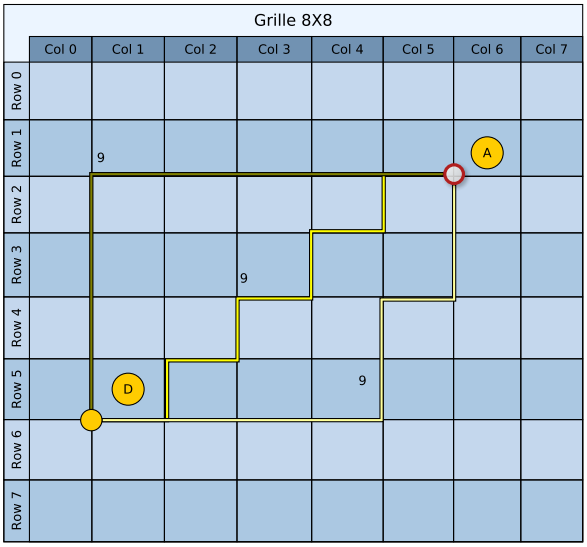
8.3.1.3 Voisinage

Pour la variable $V_{l,c}$, son voisinage[3] (cf. figure 8.4) correspond à l'ensemble noté $\mathcal{V}^d_{l,c}$ défini comme étant :

$$\mathcal{V}^d_{l,c} = \{V_{i,j}, i \in [l-d, l+d], j \in [c-d, c+d]\} \setminus \{V_{l,c}\} \tag{8.4}$$

[3]On parle ici des cellules voisines sur la grille dans le sens géométrique du terme, à ne pas confondre avec la notion de voisinage exploitée par les algorithmes de recherche locale.

FIGURE 8.3 – Illustration de la distance de *Manhattan* à parcourir entre le point de départ « D » et le point d'arrivée « A ».



où : d est un paramètre contrôlant la taille du voisinage. Cela correspond au voisinage de MOORE mais privé de son centre, c'est à dire un ensemble de cellules délimité par une distance de *Tchebychev* non nulle à partir de la cellule $V_{l,c}$.

8.3.1.4 Anneau de voisinage

Il nous arrivera également de considérer l'ensemble des cellules positionnées sur l'anneau de voisinage (cf. figure 8.5) situé à la périphérique de $\mathcal{V}^d_{l,c}$ noté $\bar{\mathcal{V}}^d_{l,c}$ tel que :

$$\bar{\mathcal{V}}^d_{l,c} = \{V_{i,j}, (|i-l| = d) \vee (|j-c| = d)\} \tag{8.5}$$

8.3.1.5 Filtrage d'un ensemble de cellules relatives à une forme urbaine

Par rapport à un ensemble de cellules donné, on veut déterminer le sous-ensemble de ces cellules qui sont affectées à une forme urbaine particulière.

Soit $\mathcal{V}$ un ensemble de cellules et f une forme urbaine donnée, $\mathcal{V}_{\triangleright f}$ est l'ensemble des éléments $V_{l,c}$ de $\mathcal{V}$ tel que $V_{l,c} = f$.

8.3.1.6 Filtrage d'un ensemble de formes urbaines relatives à un ensemble de cellules

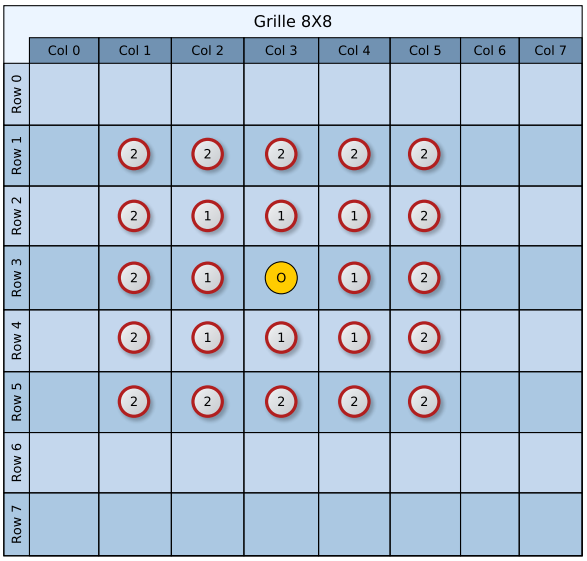
Cette fois, on dispose d'un ensemble de formes urbaines et on veut déterminer le sous-ensemble de ces formes urbaines qui sont affectées à un ensemble de cellules donné.

Soit $\mathcal{F}$ un ensemble de formes urbaines et $\mathcal{V}$ un ensemble de cellules, la notation $\mathcal{F}_{\triangleright\mathcal{V}}$ retourne les formes urbaines de $\mathcal{F}$ ayant une correspondance dans $\mathcal{V}$. Par exemple, si $\mathcal{F} = \{10, 12, 13, 14, 17, 18\}$ et $\mathcal{V} = \{V_1 = 6, V_2 = 18, V_3 = 18, V_4 = 12\}$, alors $\mathcal{F}_{\triangleright\mathcal{V}} = \{12, 18\}$.

8.3.2 Contraintes générales

Les contraintes générales s'appliquent à toutes les formes urbaines sur la totalité de la grille. Elles permettent de définir une organisation générale de la ville qui sera raffinée par des contraintes plus spécifiques.

FIGURE 8.4 – $(\mathcal{V}^2_{3,3})$ représente le voisinage de la cellule marquée par un « O » et positionnée en ligne 3, colonne 3 pour une taille de voisinage = 2 : le voisinage est constitué de toutes les cellules marquées par un « 1 » et un « 2 ».



8.3.2.1 Cardinalité liée à chaque forme urbaine

Selon certaines caractéristiques de la ville à créer (nombre d'habitants, taille, taux d'emploi, etc.), les urbanistes sont en mesure de déterminer le nombre d'occurrences de chaque forme urbaine devant apparaître dans la ville : par exemple, une ville importante doit avoir une certaine surface de parcs, une certaines surface d'industries, etc. Pour chaque forme urbaine, on a donc une contrainte de cardinalité sur l'ensemble des îlots.

8.3.2.2 Correspondance des niveaux d'intensité

Le modèle urbain fournit un niveau d'intensité donné pour chaque forme urbaine, et un niveau d'intensité pour chaque cellule de la grille. Sur la base de ces éléments, chaque assignation d'une forme urbaine à une cellule doit respecter la correspondance d'intensité entre les formes urbaines et les cellules. Les différents niveaux d'intensité sont définis par l'utilisateur sur la carte. Il s'agit d'une contrainte unaire.

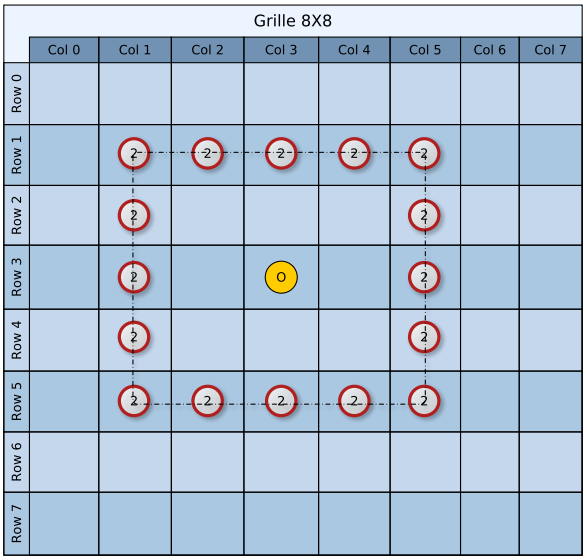
8.3.2.3 Interactions entre formes urbaines

Chaque forme urbaine a des préférences de placement en fonction de sa nature. Par exemple, les unités scolaires sont attirées par les unités résidentielles, et les unités résidentielles sont repoussées par les unités industrielles [121]. Nous modélisons ces préférences par une fonction spécifiant l'attraction (ou inversement la répulsion) de chaque forme urbaine pour une autre forme urbaine. Entre deux formes urbaines, les valeurs d'interaction possibles sont :

- 0 : double attraction,
- 10 : simple attraction,
- 20 : neutre,
- 50 : simple répulsion,
- 100 : double répulsion.

La valeur d'interaction diminue avec l'augmentation de la distance entre deux cellules (cf. figure 8.6). Les préférences d'interaction sont exprimées comme une fonction de coût. Nous notons Ω^I la matrice

FIGURE 8.5 – $(\bar{\mathcal{V}}^2_{3,3})$ représente l'anneau de voisinage de la cellule marquée par un « O » et positionnée en ligne 3, colonne 3 pour une taille de voisinage = 2 : l'anneau est constitué de toutes les cellules marquées par un « 2 ».



d'interaction, qui est une entrée du système. $\Omega^I_{p,q}$ est la valeur d'interaction entre la forme urbaine p et la forme urbaine q. Cette matrice est asymétrique, on peut donc avoir $\Omega^I_{p,q} \neq \Omega^I_{q,p}$.

Pour une forme urbaine positionnée sur la cellule $V_{l,c}$, le coût d'interaction dépend uniquement de son voisinage et de l'influence de la distance avec ses cellules voisines. Finalement, pour une cellule $V_{l,c}$, la fonction de coût relative à notre contrainte est :

$$Cost_1\left(V_{l,c}\right) = \sum_{d=1}^{\delta_1} \left(\frac{\sum_{v \in \bar{\mathcal{V}}^d_{l,c}} \left(\Omega^I_{V_{l,c},v} \right)}{\left| \bar{\mathcal{V}}^d_{l,c} \right| * d^2} \right) \tag{8.6}$$

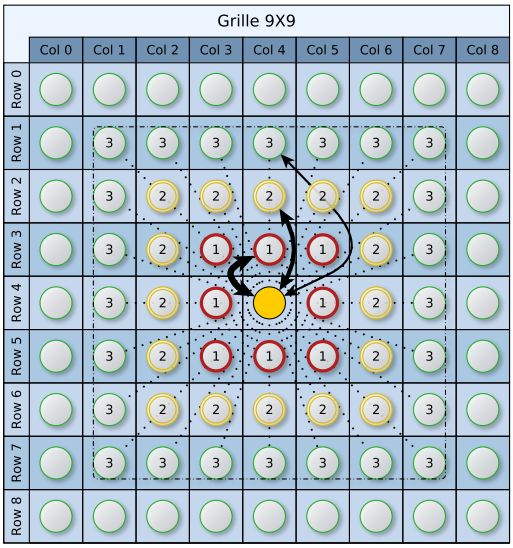
où : δ_1 est la distance d'interaction maximale (dans la suite, δ_1 est fixé à « 3 ») à considérer entre deux formes urbaines. Le coût de la cellule $V_{l,c}$ intègre les contributions de tous les anneaux compris à la distance d de $V_{l,c}$, mais ces contributions diminuent au fur et à mesure que d augmente. Pour un anneau, les contributions moyennes des cellules (situées sur l'anneau) sont divisées par un facteur de correction égale à d^2, afin d'obtenir un effet similaire à l'attraction en physique (cf. les modèles d'interaction spatiale et les modèles gravitaires 7.1.3).

Comme indiqué sur le graphique situé à droite dans la figure 8.6, les cellules en bordure de grille, notamment dans les coins, ont un voisinage limité par rapport aux cellules éloignées des bords. Pour prendre en compte cette variation de taille du voisinage, l'équation 8.6 prend en compte le nombre de cellules présentes dans chaque anneau pour définir une moyenne des interactions de la cellule concernée avec ses cellules voisines.

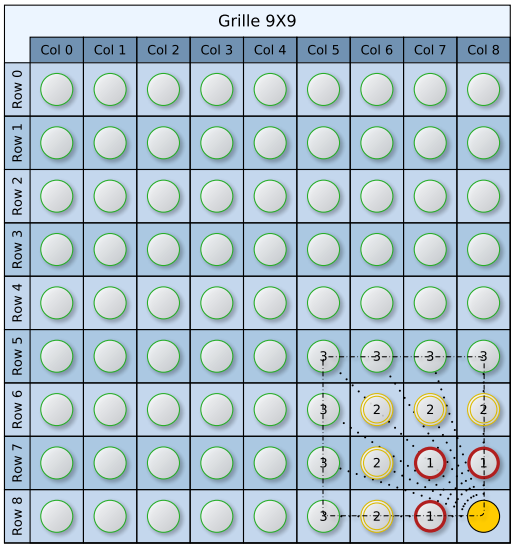
8.3.3 Contraintes spécifiques

Afin de renforcer les aspects durables de notre modèle, nous ajoutons des contraintes plus spécifiques visant à améliorer l'équité sociale, la préservation de l'environnement et la viabilité économique.

FIGURE 8.6 – Contrainte d'interaction entre formes urbaines permettant d'exprimer les préférences de positionnement des formes urbaines les unes par rapport aux autres. Chaque cellule est en interaction avec ses cellules voisines dans un voisinage limité à une taille de 3 cellules. La force d'interaction entre deux cellules diminue avec l'augmentation de la distance qui les sépare.



Interaction avec un voisinage complet



Interaction avec un voisinage réduit

8.3.3.1 Distance minimale de séparation

Cette contrainte spécifie que certaines formes urbaines doivent être éloignées les unes des autres d'une distance minimale, la distance étant exprimée en nombre de cellules (cf. figure 8.7). Par exemple, entre une maison individuelle (n° 1) et un immeuble tertiaire R+7 (n° 7), il faut respecter une distance d'au moins « 4 » cellules (i.e. trois cellules de séparation). Pour cette contrainte, nous utilisons la distance de *Tchebychev* pertinente pour contrôler la proximité de formes urbaines dans un voisinage donné.

Nous notons Ω^D la matrice de distance, avec $\Omega^D_{p,q}$ la distance minimum autorisée entre la forme urbaine p et la forme urbaine q. Cette matrice est symétrique. Par convention, $\Omega^D_{p,q} = 0$ lorsqu'il n'y a pas de contrainte de distance particulière entre p et q. Pour une forme urbaine $V_{l,c}$ localisée en (l, c), la fonction de coût liée à notre contrainte devient :

$$Cost_2\left(V_{l,c}\right) = \sum_{v \in \mathcal{V}^3_{l,c}} \left(\max\left(\Omega^D_{V_{l,c},v} - \mathcal{D}_{\text{tch}}(V_{l,c}, v), 0\right)\right) \tag{8.7}$$

Nota : On travaille ici avec une taille de voisinage limitée à « 3 » cellules car la distance minimale la plus importante relevée dans notre matrice de distance est égale à « 4 ». Donc au delà de cette limite de voisinage, il n'y a jamais de pénalité.

8.3.3.2 Regroupement

Les zones industrielles ou artisanales doivent atteindre une taille critique pour que leur construction soit économiquement viable, c'est à dire un regroupement minimum de la même forme urbaine localisée dans une même région, autrement la zone ne sera pas créée en pratique. Pour d'autres formes urbaines, en plus d'une taille critique à atteindre, on doit limiter le regroupement à une taille maximale pour favoriser leur répartition sur le territoire et indirectement leur accessibilité. C'est le cas des lycées et des hypermarchés. Les paramètres de regroupement sont déterminés par les concepteurs urbains pour chaque forme urbaine concernée par un regroupement (voir par exemple la table 8.1). D'un autre côté, bien que cela ne soit pas explicitement exprimé par les concepteurs, la zone considérée doit avoir une structure suffisamment compacte

FIGURE 8.7 – Contrainte de distance minimale de séparation entre formes urbaines.

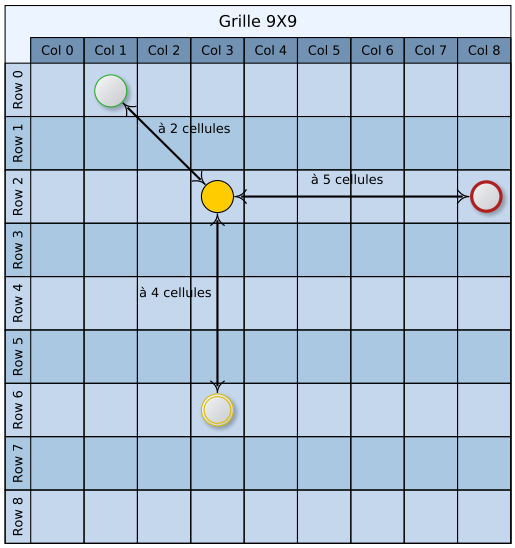


TABLE 8.1 – Paramètres des contraintes de regroupement par forme urbaine.

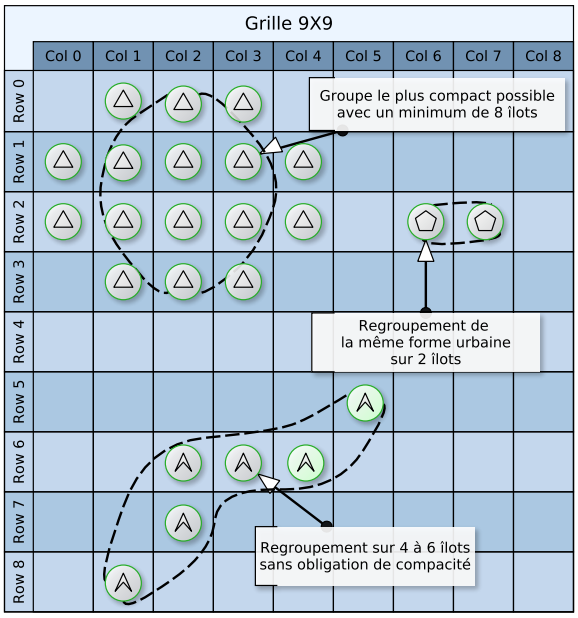
Forme urbaine	*Nombre minimum d'îlots*	*Nombre maximum d'îlots*	*Proportion d'îlots voisins identiques*
Artisanat	8	-	80 %
Industrie	42	-	80 %
Hypermarché	4	6	0 %
Lycée	2	2	0 %

pour le regroupement de certaines formes urbaines. Intuitivement, on peut penser qu'un regroupement sous la forme d'un cercle est préférable à une ligne ou un rectangle plat. Pour ces formes urbaines, on pénalise alors les cellules qui appartiennent à un groupe trop petit ou pas assez compact. On considère également qu'un même groupe peut enjamber des éléments fixes positionnés sur le territoire comme une rivière, une route ou un terrain accidenté, à condition toutefois que cette séparation ne soit pas supérieure à un îlot.

Soit les éléments suivants donnés en entrée du système :

- $\mathcal{F}_{\text{reg}}$ l'ensemble des formes urbaines à regrouper, et f une forme urbaine appartenant à cet ensemble ;
- $\text{Size}^f_{\text{min}}$ la taille minimale pour un groupe de f ;
- $\text{Size}^f_{\text{max}}$ la taille maximale pour un groupe de f ;
- $\text{Coef}^f_{\text{struct}}$ le coefficient de structure pour un groupe de f avec $0 < \text{Coef}^f_{\text{struct}} \leq 1$. Ce coefficient permet de calculer la proportion, pour une cellule donnée, de ses cellules voisines devant appartenir au même groupe (i.e. ayant la même forme urbaine). Il influence la structure géométrique des groupes : plus sa valeur est proche de « 1 », plus la structure du groupe généré sera compacte ;
- $\mathcal{F}_{\text{cont}}$ l'ensemble des formes urbaines de continuité pouvant occuper une cellule à l'intérieur d'un groupe sans provoquer la scission du groupe correspondant. $\mathcal{F}_{\text{cont}}$ est composé des formes urbaines suivantes :
  - rivière ;
  - route (résidentielle, intermédiaire, avenue) ;
  - terrain accidenté ;
  - espace naturel ;
  - voie préexistante ;

FIGURE 8.8 – Contraintes de regroupement d'une forme urbaine.



- voie de chemin de fer ;
- zone inconstructible ;

On note $G^f_{l,c}$ le groupe de cellules associé à $V_{l,c}$ avec $f = V_{l,c}$ tel que :

$$G^f_{l,c} = \begin{cases} \emptyset & \text{Si } V_{l,c} \neq f \\ V_{l,c} & \text{Si } V_{l,c} = f \\ \forall\, V_{l',c'} \in \mathcal{V}^1_{l,c}, G^f_{l',c'} & \text{Si } \left(V_{l',c'} \notin G^f_{l,c}\right) \wedge \left((V_{l',c'} = f) \vee \left((V_{l',c'} \in \mathcal{F}_{\text{cont}}) \wedge (V_{l,c} = f)\right)\right) \end{cases} \tag{8.8}$$

On note $\mathcal{C}_{l,c}$ le coût associé au groupe $G^f_{l,c}$ tel que :

$$\mathcal{C}_{l,c} = \begin{cases} \left(\text{Size}^{V_{l,c}}_{\text{min}} - \left|G^{V_{l,c}}_{l,c}\right|\right) & \text{Si Size}^{V_{l,c}}_{\text{min}} > \left|G^{V_{l,c}}_{l,c}\right| \\ \left(\left|G^{V_{l,c}}_{l,c}\right| - \text{Size}^{V_{l,c}}_{\text{max}}\right) & \text{Si Size}^{V_{l,c}}_{\text{max}} < \left|G^{V_{l,c}}_{l,c}\right| \\ 0 & \text{Sinon} \end{cases} \tag{8.9}$$

Le coût de regroupement associé à la variable $V_{l,c}$ peut alors s'écrire :

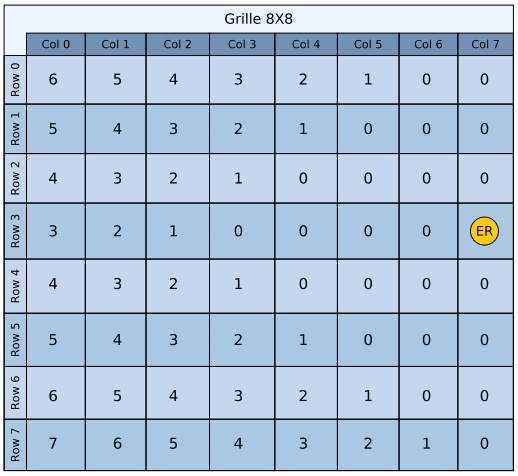
$$Cost_3\left(V_{l,c}\right) = \begin{cases} \max\left(\left(\text{Coef}^{V_{l,c}}_{\text{struct}} * \left|\mathcal{V}^1_{l,c}\right|\right) - \left|\mathcal{V}^1_{l,c_{\triangleright V_{l,c}}}\right|, 0\right) + \mathcal{C}_{l,c} & \text{Si } V_{l,c} \in \mathcal{F}_{\text{reg}} \\ 0 & \text{Sinon} \end{cases} \tag{8.10}$$

8.3.3.3 Accessibilité

Cette contrainte concerne uniquement la forme urbaine : espaces de respiration. Elle spécifie qu'en tout point habité de la ville, on doit pouvoir accéder à un espace de respiration en moins de quinze minutes à pied. Pour un piéton se déplaçant à une vitesse moyenne de 5 km/h, ce temps lui permet de parcourir une distance d'environ 1,25 km, équivalente à la longueur de 15 cellules mises bout à bout.

Nous proposons deux versions complémentaires de cette contrainte. La première version dite « *contrainte d'accessibilité globale* » pénalise les cellules habitées et non couvertes proportionnellement au dépassement

FIGURE 8.9 – Contrainte d'accessibilité globale avec une distance de couverture limitée à 4 cellules (en distance de Manhattan) : coût imputé sur chaque cellule habitée et non couverte par un espace de respiration. Dans cet exemple, la cellule marquée « ER » correspond à la seule cellule occupée par un espace de respiration.



de leur distance autorisée à n'importe quel espace de respiration. La seconde version intitulée « *contrainte d'accessibilité locale* » pénalise uniformément les cellules habitées et non couvertes par un espace de respiration.

La version globale est plus appropriée que la version locale si le nombre d'espaces de respiration à répartir sur la ville est insuffisant pour couvrir la grille entière, mais son temps de calcul est lié à la taille de la grille et peut donc devenir conséquent. En revanche, la version locale peut être calculée beaucoup plus rapidement dans un espace limité autour de chaque cellule habitée et elle est pertinente si le modèle urbain fournit suffisamment d'espaces de respiration pour couvrir toute la zone à développer.

Contrainte d'accessibilité globale À ce niveau, on mesure à quel point la contrainte est violée en quantifiant la violation au lieu de simplement l'observer. La distance entre cellules est mesurée en distance de Manhattan, pertinente pour un déplacement piétonnier en ville.

Les coûts correspondants liés à chaque cellule habitée sont illustrés sur la figure 8.9, en supposant qu'il n'y ait qu'un seul espace de respiration (repéré par « ER ») dans la grille, que toutes les autres cellules sont habitées et que la distance de couverture d'un espace de respiration, au dessous de laquelle il n'y a pas de coût, soit fixée avec une distance de Manhattan $\delta_2 = 4$ (au lieu de 15 normalement, ceci pour simplifier la représentation).

La fonction de coût associée à cette contrainte correspond alors à la somme de toutes les violations pour toutes les cellules habitées. Pour la calculer, il est nécessaire de connaître les positions des espaces de respiration dans la configuration courante, que l'on note $\text{ER}_1, \ldots, \text{ER}_m$ (en supposant qu'il y ait m espaces de respiration). $\text{ER}_1 = (l, c)$ signifie que le premier espace de respiration correspond à la variable $V_{l,c}$. On note $\mathcal{F}_{\text{hab}} = \{1, 2, 3, 4, 5\}$, l'ensemble représentant les habitations et correspondant aux formes urbaines suivantes :

- **1** maison individuelle ;
- **2** maison de ville ;
- **3** habitat intermédiaire ;
- **4** habitat collectif (R+4) ;
- **5** habitat collectif (R+7) ;

La fonction de coût associée à une cellule $V_{l,c}$ pour la contrainte d'accessibilité globale vaut :

$$Cost_4\left(V_{l,c}\right) = \begin{cases} \min_{(1\leq\alpha\leq m)}\left(\max\left(\mathcal{D}_{\text{man}}(V_{l,c}, \text{ER}_\alpha) - \delta_2, 0\right)\right) & \text{Si } V_{l,c} \in \mathcal{F}_{\text{hab}} \\ 0 & \text{Sinon} \end{cases} \tag{8.11}$$

où :

- m est le nombre d'espaces de respiration ;
- δ_2 est une constante indiquant le nombre maximum d'îlots à parcourir à pied (distance de Manhattan) pour accéder à un espace de respiration ;
- $\mathcal{F}_{\text{hab}}$ est l'ensemble représentant les habitations.

Contrainte d'accessibilité locale Dans cette version, on s'intéresse aux cellules habitées qui ne sont couvertes par aucun espace de respiration. Contrairement à la contrainte d'accessibilité globale, on observe simplement les violations sans les quantifier. Dès lors, tous les îlots habités sont pénalisés de la même manière dès qu'ils ne sont pas couverts par un espace de respiration, que l'espace de respiration le plus proche soit éloigné ou très éloigné.

Bien que la distance de Manhattan soit idéale pour mesurer le parcours d'un piéton en ville, on fait le choix de travailler ici en distance de Tchebychev pour les raisons suivantes :

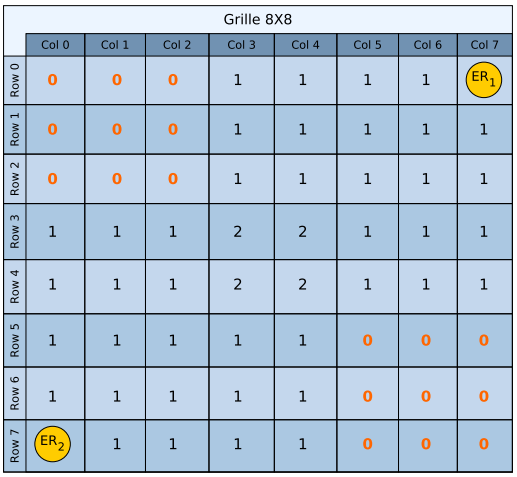
- on ne cherche pas à quantifier les violations ;
- on travaille à une échelle locale réduite autour de chaque espace de respiration ;
- le nombre d'espace de respiration est suffisant pour couvrir la ville entière (hypothèse de départ pour que cette version soit pertinente) ;
- la distance de Tchebychev nous permet d'exploiter directement le voisinage $\mathcal{V}^d_{l,c}$ pour une distance d égale à la distance de couverture imposée pour chaque espace de respiration (voir l'équation 8.12).

Profitons de cette argumentation sur le choix d'une unité de distance pour revenir aux origines de cette version locale et nous permettre une petite intrusion technique. La contrainte d'accessibilité locale est une version volontairement « dégradée » qui a été conçue pour améliorer les performances de la version globale plus élaborée et permettre ainsi un passage à l'échelle. Lors des premières expérimentations, nous avons en effet constaté qu'à partir d'une taille de grille 64×64, la version globale devenait trop pénalisante en temps de réponse pour pouvoir être utilisée dans un contexte interactif. D'un point de vue purement opérationnel et technique, l'utilisation du voisinage (représenté par $\mathcal{V}^d_{l,c}$) permettait de réutiliser (i) des procédés (algorithmes) et (ii) des données calculées par ailleurs et directement disponibles en cache (cf. section 9.3.1).

Pour un espace de respiration repéré aux coordonnées (l, c), sa zone de couverture correspond au voisinage $\mathcal{V}^{\delta_3}_{l,c}$, avec δ_3 correspondant à la distance de couverture retenue exprimée en distance de Tchebychev. Contrairement à la version globale, on récupère ici le voisinage de couverture associée à chaque espace de respiration et on cumule, au niveau de chaque cellule habitée, le nombre d'espaces de respiration qui les recouvrent. Cette notion est illustrée dans la figure 8.10, en supposant qu'il y ait deux espaces de respiration (repérés par ER_1 et ER_2) dans la grille, que toutes les autres cellules soient habitées et que $\delta_3 = 4$.

Les cellules de la grille qui contiennent le chiffre « 1 » sont couvertes par ER_1 ou ER_2, celles qui contiennent le chiffre « 2 » sont couvertes à la fois par ER_1 et ER_2. Seul le nombre de cellules habitées non couvertes, c'est à dire celles identifiées par le chiffre « 0 », sont comptabilisées pour déterminer le coût d'une configuration. La fonction de coût associée à cette contrainte correspond alors au nombre total de cellules habitées non couvertes. Pour la calculer, il est nécessaire, comme pour la version globale, de connaître les positions des espaces de respiration dans la configuration courante, que l'on notera de la même manière. La fonction de coût associée à la contrainte d'accessibilité locale pour la variable $V_{l,c}$ est :

FIGURE 8.10 – Contrainte d'accessibilité locale avec une distance de couverture limitée à 4 cellules (distance de Tchebychev) : on indique le nombre d'espaces de respiration qui couvrent chaque cellule habitée. Dans cet exemple, les cellules marquées « ER_1 » et « ER_2 » correspondent aux cellules occupées par un espace de respiration. Les cellules habitées marquées avec le chiffre « 0 » sont les cellules qui vont alimenter la fonction de coût (pénalité pour chaque cellule sans couverture).



$$Cost_4\left(V_{l,c}\right) = \begin{cases} 1 & \text{Si } \left(V_{l,c} \in \mathcal{F}_{\text{hab}}\right) \wedge \sum\limits_{1 \leq \alpha \leq m} \begin{cases} 1 & \text{Si } V_{l,c} \in \mathcal{V}_{\text{ER}_\alpha}^{\delta_3} \\ 0 & \text{Sinon} \end{cases} = 0 \\ 0 & \text{Sinon} \end{cases} \tag{8.12}$$

où :

- m est le nombre d'espaces de respiration ;
- δ_3 est une constante indiquant le nombre maximum d'îlots à parcourir à pied (distance de Tchebychev) pour accéder à un espace de respiration ;
- $\mathcal{F}_{\text{hab}}$ est l'ensemble représentant les habitations.

8.3.3.4 Séparation

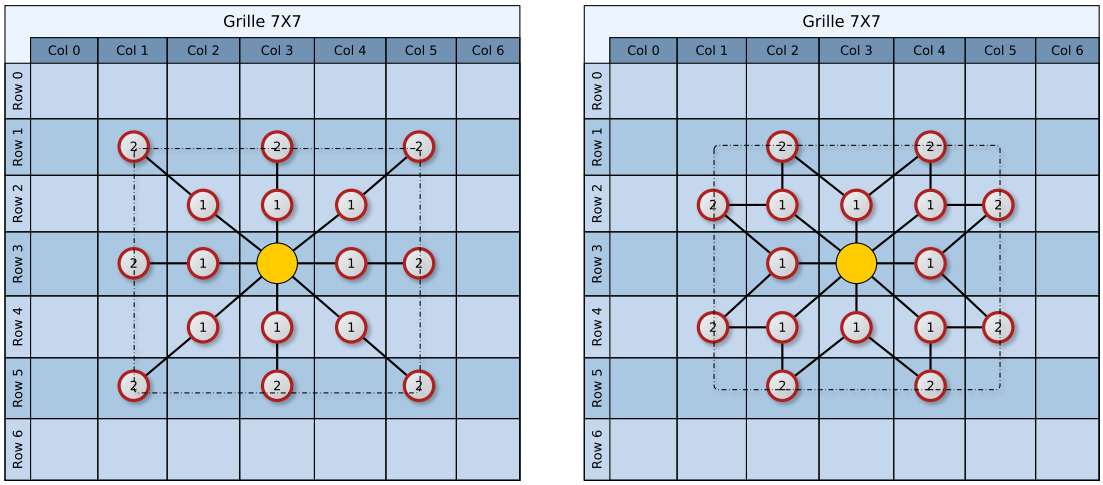
C'est une contrainte inhérente aux espaces tampons ou aux équipements bâtis qui doivent s'intercaler entre certaines formes urbaines ne pouvant pas se toucher. En effet, pour certaines formes urbaines proches les unes des autres, il est demandé qu'elles ne soient pas contiguës (qu'elles ne se touchent pas), et qu'elles soient séparées par un espace tampon ou un équipement bâti lorsqu'elles sont situées à proximité l'une de l'autre. Les formes urbaines suivantes sont considérées comme des espaces tampon :

- équipement sportif ;
- espace de respiration ;
- espace vert

Tandis que les équipements bâtis sont représentés par les formes urbaines suivantes :

- école maternelle et élémentaire ;
- collège ;
- lycée ;
- équipement administratif ;
- équipement technique.

FIGURE 8.11 – Illustration des séparations simples ou doubles entre la cellule centrale et les cellules de l'anneau périphérique situées à une distance stricte $d = 2$.



Séparation simple Double séparation

Par exemple, lorsqu'une maison individuelle et un habitat collectif R+7 se trouvent à proximité l'un de l'autre, nous devons les séparer par un espace tampon. Lorsqu'une maison individuelle est proche d'un habitat collectif R+4, il faut positionner entre les deux îlots un espace bâti.

Pour une forme urbaine repérée par $V_{l,c}$, on s'intéresse uniquement à ses variables voisines présentes dans l'ensemble $\mathcal{V}^2_{l,c}$. On note Ω^S la matrice de séparation fournie en entrée du problème. $\Omega^S_{p,q}$ fournit l'ensemble des formes urbaines autorisées pour séparer la forme urbaine p de la forme urbaine q. Soulignons que chaque élément de cette matrice correspond à un ensemble (constitué de formes urbaines), pratique suffisamment inhabituelle pour qu'elle soit précisée. Cette matrice est symétrique, on aura donc $\Omega^S_{p,q} = \Omega^S_{q,p}$. On peut avoir $\Omega^S_{p,q} = \emptyset$ lorsqu'il n'y a pas de contrainte particulière de séparation entre les deux formes urbaines p et q.

Pour chaque variable $V_{i,j}$ située dans le voisinage $\mathcal{V}^1_{l,c}$, on doit vérifier $\Omega^S_{V_{l,c},V_{i,j}} = \emptyset$. En effet, lorsqu'une séparation est imposée entre deux formes urbaines, celles-ci ne doivent pas être contiguës. D'autre part, pour chaque cellule $V_{i,j}$ située sur l'anneau de voisinage $\bar{\mathcal{V}}^2_{l,c}$, si $\Omega^S_{V_{l,c},V_{i,j}} \neq \emptyset$, il faut vérifier que les formes urbaines positionnées sur les cellules qui séparent $V_{l,c}$ et $V_{i,j}$ appartiennent bien à l'ensemble $\Omega^S_{V_{l,c},V_{i,j}}$.

La notation $\mathcal{S}(V_{l,c}, V_{i,j})$ désigne l'ensemble des formes urbaines positionnées sur les cellules qui séparent $V_{l,c}$ et $V_{i,j}$. Cet ensemble peut contenir un ou deux éléments comme le montre la figure 8.11 qui indique les variables de séparation entre la cellule centrale $V_{3,3}$ et ses cellules voisines $V_{i,j}$ situées à une distance stricte $d = 2$. Par exemple, $\mathcal{S}(V_{3,3}, V_{1,1})$ contient la forme urbaine de $V_{2,2}$ et $\mathcal{S}(V_{3,3}, V_{1,2})$ contient les formes urbaines de $V_{2,2}$ et de $V_{2,3}$.

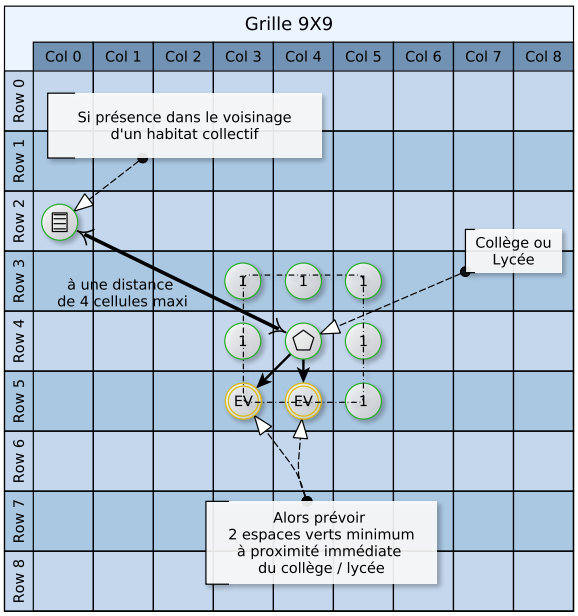
La fonction de coût pour la variable $V_{l,c}$ peut alors s'exprimer ainsi :

$$Cost_5\left(V_{l,c}\right) = \sum_{v \in \mathcal{V}^1_{l,c}} \begin{cases} 1 & \text{Si } \left(\Omega^S_{V_{l,c},v} \neq \emptyset\right) \\ 0 & \text{Sinon} \end{cases} + \sum_{v \in \bar{\mathcal{V}}^2_{l,c}} \begin{cases} \left|\mathcal{S}(V_{l,c}, v) \backslash \Omega^S_{V_{l,c},v}\right| & \text{Si } \left(\Omega^S_{V_{l,c},v} \neq \emptyset\right) \\ 0 & \text{Sinon} \end{cases} \tag{8.13}$$

8.3.3.5 Emprise au sol

Le principe est le suivant : pour des formes urbaines particulières (maison individuelle, maison de ville, habitat intermédiaire ou collectif, bâtiment tertiaire) que l'on trouve dans le voisinage proche d'un collège ou d'un lycée, il faut réserver une surface autour de l'école, en positionnant, à coté de l'édifice, un nombre plus ou moins important d'espaces verts.

FIGURE 8.12 – Emprise au sol : le nombre d'espaces verts au pied d'un collège ou d'un lycée est imposé et ce nombre dépend de la présence de certaines formes urbaines particulières à proximité de l'établissement.



Par exemple, si on ne trouve à proximité d'un collège que des maisons individuelles, des maisons de ville ou de l'habitat intermédiaire, il faut prévoir au moins un espace vert à côté de l'école. Cependant, dès qu'il y a de l'habitat collectif ou des bâtiments tertiaires aux environs du collège, nous devons prévoir une surface plus importante constituée d'au moins deux espaces verts (cf. figure 8.12). On note Ω^E la matrice qui contient le paramétrage d'emprise au sol, tel que $\Omega^E_{p,q}$ identifie le nombre minimum d'espaces verts à positionner autour de la forme urbaine p compte tenu de la proximité de la forme urbaine q. Cette matrice est asymétrique, donc $\Omega^E_{p,q}$ peut être différent de $\Omega^E_{q,p}$. On peut également avoir $\Omega^E_{p,q} = 0$ lorsqu'il n'y a pas de conditions particulières d'emprise au sol entre les deux formes urbaines p et q. Concernant la proximité liée à une forme urbaine $V_{l,c}$ de référence, seules les cellules voisines représentées par l'ensemble $\mathcal{V}^4_{l,c}$ sont considérées.

Soit :

- $\mathcal{F}_{\text{res}}$ l'ensemble des formes urbaines concernées par la réservation d'un ou de plusieurs espaces verts dans leur voisinage immédiat (i.e. collège et lycée) ;
- EV représentant la forme urbaine « espace vert »,

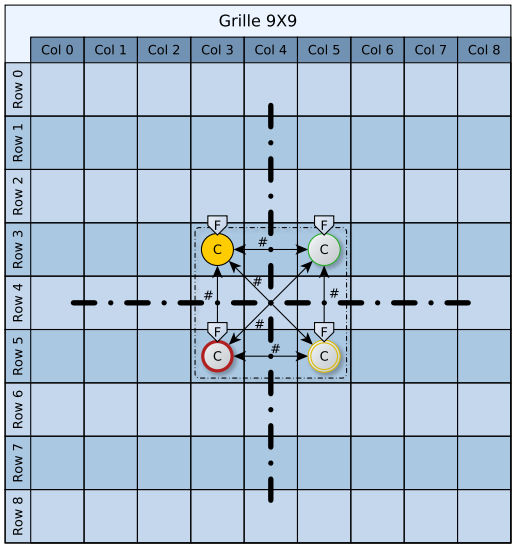
Si $V_{l,c} \in \mathcal{F}_{\text{res}}$, le coût associé à cette variable correspond au nombre d'espaces verts manquants à proximité immédiate de $V_{l,c}$:

$$Cost_6\left(V_{l,c}\right) = \begin{cases} \max\left(\left(\max_{(v \in \mathcal{V}^4_{l,c})}\left(\Omega^E_{V_{l,c},v}\right) - \left|\mathcal{V}^1_{l,c_{\triangleright \text{EV}}}\right|\right), 0\right) & \text{Si } \left(V_{l,c} \in \mathcal{F}_{\text{res}}\right) \\ 0 & \text{Sinon} \end{cases} \tag{8.14}$$

8.3.3.6 Filtrage et diversité

Cette contrainte est liée aux centralités (voir la section 8.2.1.2). Les centralités sont identifiées par des cellules spéciales marquées par l'utilisateur pour représenter le centre d'un quartier. Elles peuvent regrouper une, deux ou quatre cellules contiguës pour former une centralité plus ou moins importante, ces cellules pouvant être séparées par une route. Chaque cellule centralité est identifiée par l'utilisateur. On parlera indifféremment de « *centralité* » pour faire référence soit à une cellule marquée par l'utilisateur, soit à un groupe de cellules appartenant à la même centralité. Cette contrainte est utilisée pour (cf. figure 8.13) :

FIGURE 8.13 – Filtrage et diversité des formes urbaines sur chaque centralité : la centralité principale est constituée de quatre cellules repérées par l'utilisateur avec la lettre « C » et disposées de part et d'autre d'un axe routier important (avenues). Les formes urbaines positionnées sur ces cellules doivent être « $\neq$ » entre elles pour favoriser la diversité et un filtre « F » limite les formes urbaines pouvant prendre place sur chaque cellule.



- filtrer les formes urbaines pouvant occuper une centralité (cellule) ;
- participer à la diversité des formes urbaines localisées sur une même centralité (groupe).

Pour calculer cette contrainte, il est nécessaire de connaître l'ensemble des cellules repérées par l'utilisateur comme étant des centralités, que l'on note $\mathcal{V}_{cent} = \{V_{i,j}, \ldots, V_{k,l}\}$, $V_{i,j}$ correspondant à la première centralité et $V_{k,l}$ à la dernière. On note $\mathcal{F}_{\text{cent}} = \{10, 12, 13, 14, 17, 18\}$, l'ensemble des formes urbaines autorisées à être positionnées sur une centralité et correspondant aux formes urbaines suivantes :

- **10** : commerce et service de centre ville ;
- **12** : école maternelle et élémentaire ;
- **13** : collège ;
- **14** : lycée ;
- **17** : équipement sportif ;
- **18** : espace vert.

Pour une variable $V_{l,c}$, la centralité (groupe) qui lui est associée correspond à toutes ses cellules voisines repérées comme des centralités par $\mathcal{V}_{cent}$, ces cellules pouvant être séparées par une route. Pour prendre en compte cette séparation éventuelle et regrouper des centralités non contiguës (séparées par une route), on considère un voisinage complet $\mathcal{V}^2_{l,c}$, pour ne retenir que les cellules marquées comme des centralités. On note $\mathcal{C}_{l,c}$ la centralité (groupe) associée à la variable $V_{l,c}$ tel que :

$$\mathcal{C}_{l,c} = \left\{V_{i,j}, V_{i,j} \in \left(\left(\mathcal{V}^2_{l,c} \cup V_{l,c}\right) \cap \mathcal{V}_{cent}\right)\right\} \tag{8.15}$$

Pour une cellule $V_{l,c}$ liée à une centralité (cellule), son coût correspond à une pénalité composée des éléments suivants :

1. un coût fixe (défini à 10) si la cellule $V_{l,c}$ est associée à une forme urbaine non autorisée ;
2. le nombre des cellules de la centralité associées à une forme urbaine non autorisée ;
3. la différence entre le nombre de cellules de la centralité ($|\mathcal{C}_{l,c}|$) et la diversité des formes urbaines autorisées qui leur sont imputées $\left(\left|\mathcal{F}_{\text{cent}\triangleright\mathcal{C}_{l,c}}\right|\right)$.

Compte tenu de ces éléments, le coût global associé à la cellule $V_{l,c}$ peut s'écrire ainsi :

$$Cost_7\left(V_{l,c}\right) = \begin{cases} \begin{cases} 10 & \text{Si } V_{l,c} \notin \mathcal{F}_{\text{cent}} \\ 0 & \text{Sinon} \end{cases} + \sum_{v \in \mathcal{C}_{l,c}} \begin{cases} 1 & \text{Si } (v \notin \mathcal{F}_{\text{cent}}) \\ 0 & \text{Sinon} \end{cases} + |\mathcal{C}_{l,c}| - \left|\mathcal{F}_{\text{cent} \triangleright \mathcal{C}_{l,c}}\right| & \text{Si } V_{l,c} \in \mathcal{V}_{cent} \\ 0 & \text{Sinon} \end{cases} \tag{8.16}$$

8.4 Conclusion

« *Pour construire un modèle qui sera souvent unique en son genre, l'informaticien doit être capable d'écouter, d'aider, de faire et de refaire : il doit renoncer à venir avec ses modèles tout faits qui seraient à prendre ou à laisser* » [220]. Conformément aux préconisations de Jean-Charles POMEROL, c'est au prix d'efforts importants et de nombreuses réunions que nous sommes parvenus à réaliser avec l'aide indispensable des praticiens de la ville cette modélisation. Nous ne reprenons ici que les éléments essentiels retenus parmi une multitude de notions mises en lumière pendant des heures et des heures d'échanges passionnants et de formalisation, de nombreuses propositions de toutes parts ou d'explorations successives, parfois des abandons ou des retours en arrière. Un travail d'écoute, de compréhension et d'analyse en continu nous a permis, progressivement, de dégager et de mettre en forme des propriétés et des contraintes « *à gros grain* » caractéristiques d'une pré-programmation à la base de cette représentation formelle.

Nous nous intéressons plus particulièrement à une représentation symbolique de la ville qui permet de simplifier une réalité trop complexe pour être complètement appréhendée. Notre modèle de contrainte pour ce problème d'aménagement urbain reste volontairement simple pour qu'il puisse être compris par les décideurs, élément crucial pour qu'il soit accepté et utilisé dans la pratique. Malgré la particularité du problème, la modélisation proposée reste générique et repose sur les notions de voisinage et d'interaction entre formes urbaines, complétées par des contraintes plus spécifiques permettant de renforcer les aspects durables dans notre modèle. Dans ce cadre symbolique, la majorité des contraintes a pu être traduite en fonctions de coût pour exprimer des préférences de positionnement. Dès lors, l'objectif consiste à rechercher une configuration qui minimise un coût global traduisant le meilleur compromis entre les différentes préférences.

Ce modèle peut être résolu indifféremment par des méthodes complètes ou incomplètes. Il est prévu pour représenter des villes importantes comme l'exemple de Marne-la-vallée (8 728 hectares pour 234 644 habitants) illustré au début de ce chapitre. De plus, il sera exploité dans un cadre interactif où l'utilisateur peut interagir avec la résolution pour traduire des aspects impossibles à représenter dans le modèle. En effet, un système interactif d'aide à la décision (ou SIAD, voir le chapitre 6) doit rester un modèle inachevé [220], ce qui participe à son acceptation par les utilisateurs qui conservent une part non négligeable de liberté. Sans incertitude, la résolution du modèle pourrait être totalement robotisée sans laisser de place au processus de décision. Au contraire, la part d'incertitude qui reste dans les mains des opérateurs est une façon d'améliorer les processus de décision sans les bouleverser. C'est justement parce que l'on ne sait pas terminer le modèle qu'il est indispensable d'avoir un décideur devant l'écran pour l'achever. L'interactivité représente alors une part inexprimable de la modélisation pour des problèmes non structurés, c'est à dire impossibles à décrire entièrement.

9

Résolution initiale basée sur *Adaptive Search*

Le problème d'aménagement urbain qui nous intéresse peut être vu comme une variante du problème de positionnement d'équipements [80, 218] (Facility Location Problem, Cf. section 7.5), mais dans lequel tous les éléments urbains doivent être placés simultanément. Ce problème est fortement combinatoire [199, 200, 282]. Par conséquent, il est difficile de le résoudre lorsque le nombre de variables devient important. De plus, nous devons faire face à deux exigences spécifiques liées au contexte applicatif. Premièrement, l'algorithme doit être capable de passer à l'échelle pour se mesurer à des dimensions de villes réelles, avec une taille pouvant atteindre 10 000 cellules libres pour une surface totale de plus de $80\,km^2$ (correspondant aux éléments libres et fixes). Deuxièmement, les utilisateurs (urbanistes et décideurs) doivent pouvoir garder la main sur le système, en modifiant de façon interactive l'emplacement des éléments urbains, ceci pour répondre à leurs propres représentations et attentes.

Pour satisfaire ces deux exigences, nous avons conçu un même système qui intervient à deux étapes de résolution distinctes. Dans la première étape, qui correspond plus particulièrement à ce chapitre, le système calcule et propose une ou plusieurs bonnes solutions répondant aux contraintes urbaines. Et pour adresser le problème de complexité, nous avons développé une technique de résolution spécifique pour gérer efficacement la répartition des calculs sur plusieurs processeurs. La seconde étape est interactive et sera décrite en détail dans le prochain chapitre (cf. chapitre 10).

9.1 Choix d'une méthode de résolution

Une première tentative pour résoudre notre problème s'est appuyée sur les techniques complètes [222] de programmation par contraintes [16, 97, 9, 233]. Un tout premier prototype a été conçu avec le solveur CHOCO [147] pour une partie seulement des contraintes principales avec 17 formes urbaines différentes (au lieu de 28) :

- cardinalité liée à chaque forme urbaine ;
- interactions entre formes urbaines pour un voisinage limité à une distance maximale d'une seule cellule.

Ce prototype a été évalué sur une grille 16×16 avec un serveur de test du LINA[1]. Mais au bout de 12 jours, le traitement a dû être interrompu faute de résultats.

Dans le même contexte technique, nous avons réalisé un second prototype plus limité de placement des formes urbaines basé sur un algorithme de type glouton [85] (cf. section 5.5.1). Au lieu de répartir simul-

[1] Dell M610 quadri coeurs multithreadés (l'équivalent de 16 cœurs) Xeon E5620 à 2,40GHz, et 16 Go de RAM.

tanément toutes les formes urbaines sur la grille, cette version positionnait une à une les formes urbaines sur la ville sans remettre en cause les choix précédents. On intégrait une seule forme urbaine à la fois dans le problème et lorsque sa position optimale était trouvée (cellule), on fixait la variable qui lui était associée (devenant une constante du problème). Le processus se poursuivait alors en injectant une nouvelle forme urbaine dans le problème, en itérant jusqu'à ce que toutes les formes urbaines soient réparties sur l'espace urbain. Avec cette formule simplifiée, l'ordre dans lequel les formes urbaines sont sélectionnées joue bien entendu un rôle déterminant dans la qualité des solutions obtenues. Pour remplir une grille complète 16×16 composée de 206 variables libres au départ (les autres formes urbaines étant liées à des éléments fixes comme des routes ou des rivières), une résolution avec ce second prototype ne nécessitait plus qu'une vingtaine de minutes.

Notre problème est bien plus complexe que cette version simplifiée et sa taille typique peut atteindre 10 000 variables. De plus, une résolution avec des algorithmes gloutons ne conduit pas à des solutions de bonne qualité, ces algorithmes étant plutôt réservés à la génération de solutions initiales pouvant ensuite être améliorées. Enfin, le temps de résolution en contexte interactif doit être très rapide et ne pas dépasser quelques secondes [258].

Compte tenu de ces premiers résultats, nous avons abandonné l'idée d'une résolution avec des méthodes complètes et opté pour une méthode de résolution incomplète [2, 127] basée sur les techniques de recherche locale [177, 153, 42, 111, 112, 113, 115, 125]. Nous décrivons par la suite nos travaux réalisés sur les bases de la métaheuristique *Adaptive Search* [47, 48, 10] (cf. section 5.6.9) en commençant par la génération d'une configuration initiale.

9.2 Configuration initiale

Pour créer une configuration initiale soumise ensuite au processus de résolution, nous proposons deux heuristiques différentes pouvant être activées/désactivées selon un paramètre de l'application :

- une initialisation aléatoire simple ;
- une initialisation gloutonne aléatoire [245, 143](cf. section 5.5.2) qui cherche à positionner en priorité les formes urbaines les plus contraintes.

Dans la suite de ce chapitre, nous employons le terme « coût » pour désigner une pénalité calculée par les fonctions de coût liées à notre modélisation (voir chapitre 8) ; ce coût pouvant être associé, compte tenu du contexte d'utilisation, soit à une cellule de la grille, soit à la grille entière (coût d'une configuration), soit représenter une variation de ces coûts.

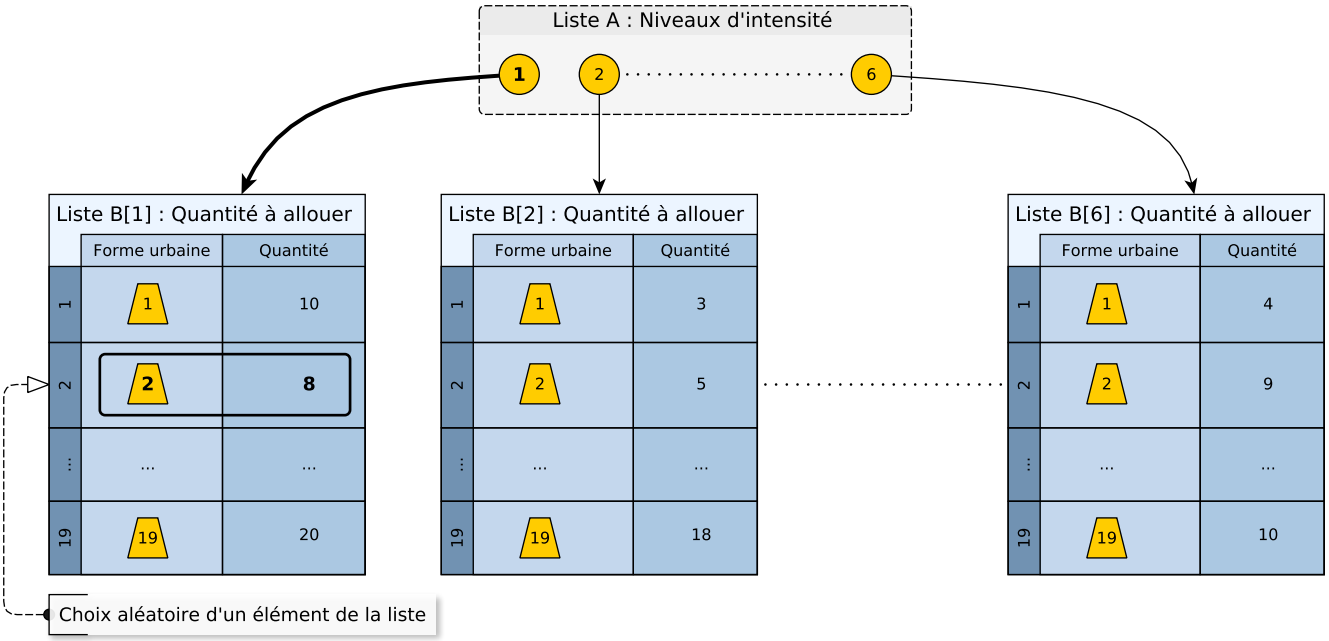
9.2.1 Initialisation aléatoire

Dans un premier temps, cette heuristique constitue une liste « A » qui répertorie par niveau d'intensité des listes « B » contenant les quantités à allouer (nombre d'îlots à occuper) par forme urbaine (cf. figure 9.1). Par exemple, la liste « B[1] » indique qu'il y a 10 maisons individuelles (forme urbaine n° 1) à répartir sur des cellules d'intensité 1 ; 8 maisons de ville (forme urbaine n° 2) à répartir sur des cellules d'intensité 1 ; et 20 espaces de respiration (forme urbaine 19) à répartir sur des cellules d'intensité 1 ; etc.

L'heuristique parcourt ensuite la grille en sélectionnant une à une chaque cellule libre. Pour la cellule courante, elle récupère son niveau d'intensité et recherche, à partir de la liste « A », la liste « B » associée à ce niveau d'intensité. Si il n'y a plus d'élément disponible dans la liste « B » trouvée, une erreur d'incohérence est émise indiquant qu'il n'y a pas suffisamment de formes urbaines allouées pour ce niveau d'intensité. Dans le cas contraire, un élément de la liste « B » est choisi aléatoirement. On récupère la forme urbaine correspondante et on l'assigne à la cellule courante. La quantité à allouer liée à cette forme urbaine est décrémentée et lorsqu'elle devient nulle, l'élément correspondant de la liste « B » est supprimé.

A la fin du processus, toutes les listes « B » doivent être vides. Dans le cas contraire, il s'agit d'une seconde incohérence et un message d'anomalie est émis indiquant qu'il y a trop de formes urbaines à

FIGURE 9.1 – Initialisation aléatoire : structures de données utilisées pour créer une solution initiale.



allouer par rapport au nombre de cellules libres.

9.2.2 Initialisation gloutonne aléatoire

Cette initialisation repose sur un paramétrage de l'application (cf. table 9.2) qui définit pour chaque forme urbaine :

- la priorité de placement (première colonne représentant le classement) ;
- le mode de sélection d'une cellule (dernière colonne « random » avec la valeur vrai ou faux).

Le classement représente la difficulté de placement de chaque forme urbaine. Il cherche à traduire le poids des contraintes qui pèse sur chaque forme urbaine. En effet, par rapport aux contraintes spécifiques, seules quelques formes urbaines sont concernées : certaines peuvent donc être plus contraintes que d'autres non concernées par les contraintes spécifiques. L'idée générale consiste alors à positionner en priorité sur la grille les formes urbaines pour lesquelles il y a un maximum de contraintes qui s'exercent.

Le mode de sélection d'une cellule indique comment on doit, pour une forme urbaine donnée, choisir une cellule parmi les cellules libres et disponibles dans la grille (i.e. sans affectation de forme urbaine). Si « Random = vrai », on choisit une cellule au hasard sinon on choisit la cellule qui génère le coût d'affectation minimum.

9.2.2.1 Fonctionnement de l'heuristique

L'heuristique considère une à une chaque catégorie de forme urbaine en fonction du classement prioritaire spécifié par le paramétrage (industrie, . . . , espace de respiration). Pour la forme urbaine courante, on récupère la liste « B » qui lui est associée et qui contient les quantités à allouer de cette forme urbaine, ces quantités étant ventilées par niveau d'intensité (cf. figure 9.3). Chaque élément présent dans la liste « B » sélectionnée vient s'insérer dans une liste « C » temporaire qui contient autant d'éléments qu'il y a d'allocations prévues, chaque élément représentant une affectation à prévoir pour le niveau d'intensité spécifié.

Tant que la liste temporaire « C » n'est pas vide, on choisit un élément aléatoire dans cette liste (une intensité) que l'on retire de la liste. A ce stade, on dispose d'une allocation à réaliser pour : (i) une forme

FIGURE 9.2 – Initialisation gloutonne : paramétrage, par forme urbaine, de la priorité de placement et du mode de sélection d'une cellule. On commence par positionner l'industrie (n° 1) et on termine par le placement des espaces de respiration (n° 19). La colonne « random » indique si on doit choisir une cellule disponible au hasard ou si on choisit celle qui génère un coût d'affectation minimum.

N°	Forme urbaine	Random
1	Industrie	faux
2	Parc d'entreprises	faux
3	Collège	faux
4	Lycée	faux
5	Maison individuelle	vrai
6	Maison de ville	vrai
7	Habitat intermédiaire	faux
8	Bureau, grand immeuble (R+7)	faux
9	École maternelle et élémentaire	faux
10	Équipement administratif	faux

N°	Forme urbaine	Random
11	Équipement technique	faux
12	Équipement sportif	faux
13	Habitat collectif (R+7)	faux
14	Bureau, petit immeuble (R+4)	faux
15	Habitat collectif (R+4)	faux
16	Commerce et service de centre ville	faux
17	Hypermarché	faux
18	Espace vert	faux
19	Espace de respiration	faux

urbaine prioritaire et (ii) une intensité tirée aléatoirement. On récupère alors la liste « D » des cellules libres, et sans affectation, pour ce niveau d'intensité. Si la liste « D » est vide, une anomalie est émise indiquant qu'il n'y a pas assez de cellules disponibles par rapport au nombre de formes urbaines à ventiler sur la ville. Dans le cas inverse, on détermine la cellule disponible qui recevra la forme urbaine. Pour cela, deux possibilités se présentent en fonction du « *mode de sélection d'une cellule* » (colonne « random ») paramétré pour la forme urbaine :

- soit on sélectionne un élément de façon aléatoire dans la liste « D » (une cellule) ;
- soit on recherche, parmi toutes les cellules de la liste « D », la cellule avec le coût le plus faible.

Pour identifier la cellule la moins coûteuse, on évalue pour chaque cellule de la liste « D » le coût que représenterait l'affectation de la forme urbaine courante et on retient la cellule avec le coût minimum (si il y a plusieurs cellules avec le même coût minimum, on fait un tirage aléatoire parmi ces cellules).

Pour la cellule retenue (quelle que soit la valeur du paramètre *mode aléatoire*), on lui attribue la forme urbaine courante et on retire cette cellule de la liste « D » des cellules libres pour le niveau d'intensité.

9.2.2.2 Évaluation du coût lié à une affectation

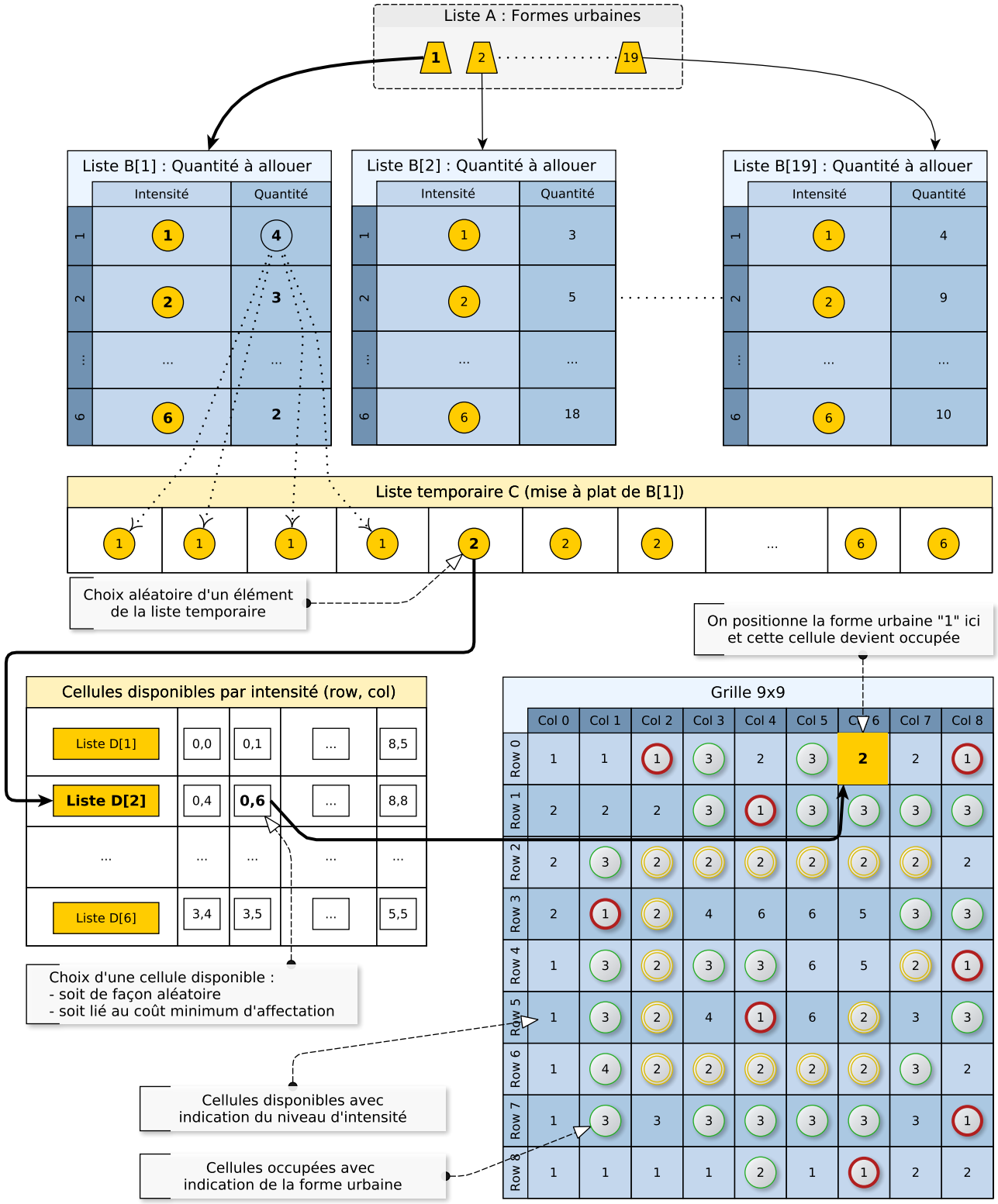
Nous verrons que notre algorithme d'optimisation (cf. section 9.3) réalise une série de permutations (i.e. échanges des valeurs entre deux variables) afin d'améliorer pas à pas le coût global d'une solution de départ. Or, la procédure gloutonne d'initialisation cherche à positionner à chaque itération une forme urbaine donnée à moindre coût sur une cellule libre, sans passer par ce mécanisme des permutations.

Afin de mutualiser ces procédures, notamment les fonctions de calcul incrémental décrites en section 9.3.1, nous avons introduit la forme urbaine « *non défini* » portant le code « 0 » qui occupe initialement toutes les cellules libres de la grille, aucune contrainte n'étant définie sur cette forme urbaine. On cherche alors à évaluer le remplacement de la forme urbaine « non défini » de chaque cellule libre par une autre forme urbaine à positionner, cette opération de placement devenant pour une large partie similaire à une permutation.

9.3 Algorithme séquentiel

Nous avons retenu la méthode *Adaptive Search* (AS) qui a prouvé son efficacité sur des instances variées et de grandes tailles [47]. Cette méthode nous permet d'améliorer, tant que c'est possible, une solution

FIGURE 9.3 – Initialisation gloutonne : structures de données utilisées pour créer une solution initiale.



courante par une succession de permutations (échange des valeurs entre deux variables) dans le but d'approcher le plus rapidement possible une solution optimale. En plus d'exploiter la fonction de coût global, cette métaheuristique tire avantage de la structure du problème en termes de contraintes et de variables pour guider la recherche.

Pour chaque contrainte, une fonction d'erreur doit être définie afin de fournir, pour chaque tuple "va-

Algorithme 11 : Notre algorithme séquentiel dérivé de la méthode Adaptive Search de base

```
1  /* Paramètres :
     maxRestart est le nombre maximum de réinitialisations partielles de l'algorithme
     f est la fonction de coût globale, f_i est la fonction de coût pour la variable V_i
     s est la configuration courante
     best est la meilleure configuration trouvée
     T est la liste des candidats bannis
     j est l'index de la variable avec le pire coût (candidat) */
2  best ← s ← configuration initiale ;
3  while (maxRestart non atteint) do
4      T ← ∅ ;
5      while (T ne contient pas la totalité des variables) do
6          forall (i tel que V_i ∉ T) do
7              Calculer f_i(s) ;
8          Sélectionner V_j une variable pour laquelle f_j(s) est maximum (avec V_j ∉ T) ;
9          Calculer le coût f des configurations voisines obtenues à partir de s en échangeant V_j avec
           une autre variable ;
10         Sélectionner s' la configuration pour laquelle f(s') est minimum;
11         if (s' peut améliorer la solution courante s) then
12             Modifier T en supprimant sa variable la plus ancienne ;
13             s ← s';
           else
14             T ← T ∪ {V_j};
15     if (s est meilleure que best) then
16         best ← s;
17     Réinitialiser un certain pourcentage de variables dans s en réalisant des permutations aléatoires ;
18 return best
```

riable | valeur", une indication sur son degré de violation [99, 272]. AS procède à une réparation itérative sur la base des erreurs liées aux variables et aux contraintes, en cherchant à réduire l'erreur sur la variable ayant le coût le plus élevé. Pour cela, on calcule la fonction d'erreur pour chaque contrainte, puis on combine pour chaque variable les erreurs de toutes les contraintes où elles apparaissent, ce qui permet de projeter les erreurs des contraintes sur les variables correspondantes. La procédure qui permet de transférer les erreurs des contraintes sur les variables est problème-dépendant.

On peut retrouver une description détaillée de la métaheuristique AS dans sa version générale en section 5.6.9. Pour ce qui nous concerne, nous proposons dans un premier temps une implémentation séquentielle (cf. algorithme 11) directement dérivée de la version originale.

Notre algorithme démarre à partir d'une configuration initiale générée soit par une affectation aléatoire, soit par une affectation gloutonne aléatoire : un paramètre permet d'identifier le mode d'initialisation à utiliser (cf. section 9.2). À ce stade, nous nous assurons que l'assignation initiale respecte à la fois le niveau d'intensité de chaque cellule et le nombre donné de cellules de chaque forme urbaine (de façon analogue à la contrainte unaire de correspondance des niveaux d'intensité et de cardinalité liée à chaque forme urbaine).

L'algorithme réalise alors une variante de la méthode de réparation itérative (Iterative Repair), basée sur les données d'erreur relatives aux variables et aux contraintes, cherchant à réduire à chaque itération l'erreur sur la variable de pire coût [192]. Notre modélisation (cf. chapitre 8) permet de calculer la fonction de coût de chaque contrainte combinée à chaque variable pour toutes les contraintes où elle apparaît, projetant directement les erreurs identifiées par les contraintes sur chaque variable. Pour obtenir un coût total par

variable, nous pondérons les coûts des contraintes de sorte que les contraintes peuvent avoir une priorité différente.

À chaque itération, la variable avec l'erreur la plus importante est désignée comme « *le candidat* » choisi pour être à l'origine d'une permutation, c'est à dire impliqué dans un échange de sa valeur avec une autre variable. Dans cette étape, on considère toutes les permutations possibles impliquant le « *candidat* » en choisissant la meilleure permutation, c'est à dire la permutation qui permettra de minimiser le plus possible le coût global de la configuration courante. Cependant, il y a des restrictions à l'ensemble des échanges considérés : nous pouvons échanger les valeurs de deux cellules uniquement si elles possèdent des formes urbaines différentes (sinon l'échange est neutre) et si elles sont affectées au même niveau d'intensité, ceci afin de satisfaire la contrainte d'intensité à tout moment. Cette politique liée aux permutations assure que la contrainte de correspondance des niveaux d'intensité est maintenue satisfaite à chaque itération, tout comme la contrainte de cardinalité des formes urbaines.

L'algorithme utilise également une mémoire dynamique de court terme dans l'esprit de la recherche tabou [111, 112, 113, 115, 125], que nous intitulerons « *liste des candidats bannis* ». Notons que seuls les *candidats* incapables d'améliorer le coût global sont marqués bannis (aucune permutation avec une autre variable ne permet de diminuer le coût global de la configuration courante). Tant qu'une variable appartient à la *liste des candidats bannis*, elle ne peut pas être considérée comme un *candidat* pour les itérations suivantes.

Lorsque l'algorithme est bloqué car toutes les variables sont bannies, il redémarre en permutant sur la configuration courante un pourcentage donné de variables choisies aléatoirement. Néanmoins, avant de réaliser une permutation à partir de deux variables choisies au hasard, le système vérifie que l'échange des valeurs entre les deux variables soit autorisée (i.e. même intensité, formes urbaines différentes, ...). Le nombre de relances et de variables à réinitialiser dépendent de paramètres configurables.

Dans les sections suivantes, nous commençons par décrire une série d'optimisations mises en œuvre pour améliorer de façon significative la vitesse des calculs avant d'étudier en détail le fonctionnement de la *liste des candidats bannis*. En vue de distribuer efficacement l'algorithme AS et de pouvoir comparer les gains obtenus avec la version séquentielle, nous introduisons dès à présent deux nouvelles fonctionnalités : le mode « *multi-candidats* » et le mode « *multi-permutations* ».

9.3.1 Cache de données et calcul incrémental

Pour améliorer la vitesse des calculs, nous utilisons différents caches de données pour toutes les informations qui nécessitent un accès fréquent, les données mémorisées pour chaque contrainte étant problème-dépendant. À titre d'exemple, on mémorise les éléments suivants :

- objets généraux pouvant être réutilisés fréquemment sans devoir être recréés (e.g. positions, cellules, candidats, permutations, configurations, ...) ;
- coût global de la configuration courante ;
- mesure du coût d'accessibilité sur chaque cellule habitée pour chaque point d'origine (cellule occupée par un espace de respiration) pour la contrainte d'accessibilité globale ;
- nombre d'espaces de respiration qui recouvre chaque cellule habitée pour la contrainte d'accessibilité locale ;
- constitution des groupes (zone industrielle, artisanale, etc.) avec leur structure et les coûts associés ;
- les voisinages de chaque cellule pour la contrainte d'interactions entre formes urbaines ;
- ....

A chaque fois qu'une permutation est appliquée, un mécanisme générique associé à chaque contrainte permet de supprimer le cache de données dépendant des deux cellules concernées, de sorte qu'à la prochaine interrogation des données en cache, ces informations soient recalculées localement pour être à nouveau disponibles.

Pour évaluer l'impact d'une permutation, nous déterminons également tous les coûts de chaque contrainte de façon incrémentale, en calculant uniquement la différence de coût induite par la permutation sur une configuration donnée (cf. sections 5.6.3.4 et 7.5.2.2). Néanmoins, ce coût incrémental peut être déterminé pour une permutation potentielle sans devoir modifier la configuration, ce qui est très utile, notamment vis à vis du cache de données qui peut ainsi rester intact. Ceci est possible en considérant un objet spécial nommé *swap* transmis systématiquement à la méthode chargée de récupérer la forme urbaine d'une cellule pour une configuration donnée (voir l'algorithme 12).

Algorithme 12 : Fonction permettant de récupérer la forme urbaine d'une configuration à la position (row, col) en considérant un composant de permutation noté *swap*.

```
1 if (swap) then
2     if ((row = swap.fromRow) et (col = swap.fromCol)) then
3         return getUrbanFrom(swap.toRow, swap.toCol) ;
4     if ((row = swap.toRow) et (col = swap.toCol)) then
5         return getUrbanFrom(swap.fromRow, swap.fromCol) ;
6 return getUrbanFrom(row, col) ;
```

9.3.2 Liste des candidats bannis

Le fonctionnement de la liste des candidats bannis est inspiré de la liste tabou (voir descriptif de la recherche tabou en section 5.6.8). La liste des candidats bannis permet d'éliminer temporairement du processus de recherche les « *mauvais* » candidats. Les mauvais candidats sont des variables qui ont eu les coûts les plus importants à un moment donné mais qui ont néanmoins été incapables de participer, en lien avec une autre variable, à une permutation permettant d'améliorer la solution courante du moment (en diminuant son coût global). Cette liste évite donc à l'algorithme de boucler sur un mauvais candidat.

Cependant, ce n'est pas sa seule utilité : elle permet également d'améliorer les performances du système. En éliminant temporairement un mauvais candidat du processus de recherche, on fait l'hypothèse suivante : *un candidat identifié comme mauvais à un moment donné risque de rester mauvais un certain temps, ce délai (évalué en nombre d'itérations) s'allongeant au fur et à mesure que le processus de résolution progresse*. En retirant momentanément du circuit ces variables, on évite à l'algorithme d'évaluer inutilement une série de permutations générées à partir de candidats potentiellement défaillants. On réduit ainsi la complexité de la métaheuristique grâce à une réduction de la taille du voisinage à évaluer à chaque itération et on améliore de fait sa rapidité.

Il s'agit d'une liste dynamique de type FIFO[2] dont la dimension augmente avec le temps. Lorsqu'elle est pleine (i.e nombre d'éléments présents dans la liste = dimension courante de la liste), l'ajout d'un nouvel élément provoque la sortie du plus ancien. Sa dimension initiale est déterminée en pourcentage par rapport au nombre de variables libres de l'instance. Typiquement, ce pourcentage peut être fixé à 25 %. Ensuite, sa dimension va évoluer par palier jusqu'à atteindre le nombre total de variables libres de l'instance. A chaque palier, la dimension de la liste progresse d'un certain pourcentage (e.g. +25 %) par rapport à sa dimension courante. Reste à déterminer le moment où interviennent ces paliers ?

La dimension de la liste reste stable tant que l'algorithme peut identifier un bon candidat sur une itération (i.e un candidat capable de produire au moins une permutation qui améliore la solution courante) ou tant que la liste n'est pas pleine. Si à l'issue d'une itération donnée le candidat sélectionné s'avère mauvais[3] et si la liste est pleine, alors un palier d'extension est sollicité par la métaheuristique pour tenter d'agrandir

[2] First In, First Out

[3] Dans le mode multi-candidats (voir la section 9.3.3), lire : « si tous les candidats sélectionnés s'avèrent mauvais ».

la dimension de la liste. Dans cette situation, on considère en effet que la capacité de la liste n'est plus suffisante pour contenir l'ensemble des mauvais candidats et qu'il faut l'agrandir dans la limite de sa capacité maximale autorisée, égale au nombre de variables libres.

Lorsque la liste a atteint sa capacité maximale et qu'elle est pleine, plus aucun candidat ne peut être trouvé car ils sont tous bannis. Dans ce cas de figure extrême, deux possibilités se présentent en fonction des paramètres de configuration[4] retenus :

- soit une perturbation de la solution courante est appliquée (cf. algorithme 11, lignes 3, 5 et 17) avant de relancer la résolution ;
- soit on enchaîne directement sur une nouvelle itération après avoir vidé la liste de son contenu (suppression de tous les candidats bannis) pour donner la possibilité à certains candidats précédemment bannis de se racheter ; ce procédé étant similaire au mécanisme d'*aspiration* [66, 131] décrit pour la méthode tabou [111] (cf. section 5.6.8).

Dans les deux cas, l'opération en question peut être répétée dans la limite d'un nombre d'occurrences paramétrable.

9.3.3 Mode multi-candidats

Dans l'algorithme général AS, une seule variable à la fois peut être désignée comme *candidat* pour rechercher une permutation en lien avec les autres variables, qui puisse minimiser le coût global de la configuration courante. Nous introduisons un mode multi-candidats où un ensemble restreint de variables associées aux coûts les plus importants d'une même configuration sont sélectionnées pour rechercher la meilleure permutation, le nombre de candidats à considérer étant un paramètre de l'application (cf. algorithme 13).

Un composant logiciel dédié a été conçu pour maintenir une liste de variables triée par ordre décroissant de coût et limitée en nombre d'éléments. Lorsque la liste est pleine, l'introduction d'un nouvel élément permet soit de l'intégrer en retirant l'élément de plus faible coût, soit de ne rien faire si le nouvel élément a un coût inférieur ou égal au dernier élément de la liste.

Dans un premier temps, on parcourt séquentiellement la grille en vérifiant que la variable courante ne soit ni fixe, ni bannie. On récupère son coût disponible en cache et on l'injecte dans ce composant. Lorsque toutes les variables ont été évaluées, on dispose d'une liste avec les meilleurs candidats de la configuration courante. Il reste alors, dans un deuxième temps, à évaluer chaque candidat avec la totalité des autres variables libres pour trouver la meilleure permutation. Au cours de cette procédure, si on ne trouve aucune permutation favorable pour un candidat donné, le candidat en question est inséré dans la liste des candidats bannis. Si on ne trouve aucune permutation favorable pour l'ensemble des candidats, alors on vérifie l'état de la liste des candidats bannis. Si elle n'est pas pleine ou si elle peut encore être agrandie, on relance la procédure courante (recherche des meilleurs candidats et de la meilleure permutation). Dans le cas contraire, soit on vide la liste des candidats bannis et on relance la procédure courante, soit la procédure se termine.

Pour justifier ce mode de fonctionnement, il faut prendre en considération les deux remarques suivantes qui sont complémentaires. Premièrement, notre algorithme cherche à se diriger le plus rapidement possible vers une solution optimale en réparant au mieux, à chaque itération, la configuration courante. Or, la plus mauvaise variable peut ne pas être celle qui conduira à la meilleure permutation (i.e. la meilleure réparation). Chercher la meilleure permutation en se basant sur plusieurs candidats augmente nos chances de trouver la meilleure permutation liée à la configuration courante. Toutefois, cette recherche impliquant plusieurs candidats, le nombre de permutations à évaluer devient supérieur, ce qui occasionne un temps de traitement supérieur. Deuxièmement, dans un mode distribué, la recherche de la meilleure permutation pour plusieurs candidats peut être réalisée en parallèle, chaque processus évaluant les permutations d'un candidat. Dans ce cas, le temps de calcul global pour tous les candidats reste identique à l'évaluation d'un seul candidat, mais la chance de trouver une permutation de meilleure qualité devient supérieure.

[4] Ne pas confondre la configuration qui regroupe l'ensemble des paramètres qui conditionnent le fonctionnement du solveur avec la notion de configuration équivalente à une affectation.

Algorithme 13 : Mode multi-candidats

/* Paramètres donnés en entrée :
- $maxCandidates$ est le nombre maximum de meilleurs candidats à sélectionner
- s est la configuration courante
- $f_i(s)$ est la fonction de coût de la variable V_i pour la configuration s
- T est la liste des candidats bannis
- EVAL(s, V_i, V_j) est la fonction qui évalue sur la configuration s le gain pour un échange des valeurs entre les deux variables V_i et V_j
- VERIFIER(s, T) est la fonction qui gère T si aucune permutation gagnante n'existe pour s */

/* Variables de travail :
- $bestCandidates$ est la liste triée (par ordre de coût décroissant) des meilleurs candidats
- $metCandidates$ est la liste des meilleurs candidats déjà traités
- $\triangle_{cost}$ est le gain pouvant être apporté à s en échangeant les valeurs de deux variables
- $bestDeltaCost$ est le meilleur gain qu'un candidat peut apporter à s en échangeant sa valeur avec celle d'une autre variable
- $improvement$ est un booléen à $true$ si au moins une permutation permet d'améliorer le coût de s
- $worstCost$ est le coût du candidat de $bestCandidates$ le moins bien classé (dernier de la liste) */

$worstCost \leftarrow 0$;
$bestCandidates \leftarrow \emptyset$;
forall *(i tel que : $V_i \notin T$ et V_i libre)* **do**
    Récupérer la valeur de $f_i(s)$ si présente en cache ou la recalculer ;
    if *($|bestCandidates| = maxCandidates$)* **then**
        if *($f_i(s) \leq worstCost$)* **then**
            Ne rien faire et passer à l'indice i suivant ;
        else
            Supprimer le dernier élément de $bestCandidates$;
    $bestCandidates \leftarrow bestCandidates \cup V_i$;
    if *($|bestCandidates| = maxCandidates$)* **then**
        $worstCost \leftarrow$ coût du dernier élément de $bestCandidates$;
$metCandidates \leftarrow \emptyset$;
$improvement \leftarrow false$;
forall *(i tel que $V_i \in bestCandidates$)* **do**
    $metCandidates \leftarrow metCandidates \cup V_i$;
    $bestDeltaCost \leftarrow 0$;
    forall *(j tel que : $V_j \notin T$ et $V_j \notin metCandidates$ et V_j libre)* **do**
        $\triangle_{cost} \leftarrow$ EVAL(s, V_i, V_j) ;
        if *($\triangle_{cost} < bestDeltaCost$)* **then**
            $bestDeltaCost \leftarrow \triangle_{cost}$;
    if *($bestDeltaCost = 0$)* **then**
        $T \leftarrow T \cup V_i$;
    else
        $improvement \leftarrow true$;
if *(! $improvement$)* **then**
    VERIFIER(s, T) ;

9.3.4 Mode multi-permutations

Au lieu de travailler avec la meilleure permutation issue d'un ou de plusieurs candidats (selon que le mode multi-candidats soit actif ou non), nous proposons de mémoriser dans une liste les meilleures permutations issues des différents candidats à considérer. Cette liste est triée par ordre décroissant du gain[5] lié à chaque permutation et sa taille est limitée (paramétrable), de sorte qu'on ne conserve qu'un nombre restreint de permutations, ces permutations étant considérées comme les meilleures pour la configuration courante.

Ce fonctionnement ne nécessite pas un temps de calcul supplémentaire significatif car ces permutations doivent de toute façon être évaluées, que le mode multi-permutations soit actif ou non. Ce mécanisme introduit simplement une empreinte mémoire additionnelle liée à la liste, mais sa taille reste contrôlée et limitée. Lorsque la liste est renseignée avec les meilleures permutations identifiées pour les différents candidats, nous appliquons séquentiellement sur la configuration courante toutes les permutations mémorisées dans la liste, sous certaines conditions. Typiquement, réaliser une première permutation peut fortement impacter les scores obtenus sur les permutations suivantes de la liste. Pour qu'une permutation reste valide et qu'elle soit appliquée, elle doit donc toujours produire un bénéfice qui soit au moins égal à un certain ratio (paramétrable) de son profit initial. Lors de ce processus séquentiel, nous garantissons uniquement l'application du premier élément de la liste qui correspond à la meilleure permutation, les permutations suivantes pouvant devenir obsolètes.

Notons que pour revenir au fonctionnement général AS, il suffit de limiter à un seul le nombre maximum de candidats et le nombre maximum de permutations, ce qui revient à désactiver ces deux modes.

En mode mono-candidat et mono-permutation de l'algorithme, des résultats liés à la recherche de la meilleure permutation se trouvent gaspillés à chaque itération. Bien que les permutations appliquées en rafale en mode multi-permutations puissent ne pas être les meilleures lorsqu'elles sont appliquées séquentiellement, plusieurs d'entre elles peuvent néanmoins améliorer le coût de la solution courante. De plus, travailler avec différents candidats permet de diversifier l'origine des permutations à appliquer en cascade, minimisant ainsi l'impact d'une permutation sur les suivantes. Dans la version distribuée de l'algorithme, ces deux mécanismes conjoints (multi-candidats et multi-permutations) sont supposés tirer profit des travaux réalisés par les différents processus parallèles sans nécessiter un effort de calcul additionnel.

9.4 Version distribuée de l'algorithme

Pour faire face à la complexité des problèmes de grande taille, nous proposons de distribuer l'algorithme sur une grille de calcul [58, 266]. Un système basé sur une marche multiple parallèle [138, 205] (« *multi-walk* ») dont les principes sont décrits en section 5.7.3 a déjà été proposé pour AS [40, 75], avec de bonnes accélérations (speed-ups) sur des problèmes classiques. Toutefois, entre plusieurs exécutions de notre version séquentielle sur des instances identiques du problème d'aménagement urbain, nous relevons de faibles écarts de temps entre les différentes résolutions ; la même remarque pouvant être faite sur la qualité des résultats obtenus. Dans notre situation, ces informations indiquent qu'il sera difficile pour une approche distribuée basée sur une marche multiple de diminuer significativement les temps de résolution.
Compte tenu de ces éléments, nous choisissons un système maître-esclave pour distribuer l'évaluation des voisins : à chaque itération, nous parallélisons la recherche des meilleures permutations [266, 10]. Ce principe rappelle le fonctionnement « Évaluation concurrente des voisins » décrit en section 5.7.1, mais avec des processus esclaves qui disposent ici d'une plus large autonomie. En fait, le fonctionnement retenu se situe à mi-chemin entre une évaluation distribuée des voisins et le principe de décomposition de domaine [227] (cf. section 5.7.2) : nous parlerons de décomposition de variable pour faire référence à notre approche.

Bien qu'il ne réalise pas lui même l'ensemble des opérations habituelles, le processus maître fonctionne de façon analogue au mode séquentiel multi-candidats, multi-permutations.

[5] Comme on mémorise des valeurs négatives dans cette liste (variations sur le coût global de la configuration courante engendrées par les permutations bénéfiques), elle est triée par ordre croissant.

Algorithme 14 : Mode multi-permutations

/* Paramètres donnés en entrée :
- $maxPermutations$ est le nombre maximum de meilleures permutations à sélectionner
- s est la configuration courante
- EVAL(s, V_i, V_j) est la fonction qui évalue sur la configuration s le gain pour un échange des valeurs entre les deux variables V_i et V_j et qui mémorise les meilleures permutations
- APPLY(s) est la fonction qui applique en série les meilleures permutations trouvées
- $maxDeviation$ est le % maximum de déviation autorisé sur le gain initial pour garder une permutation */

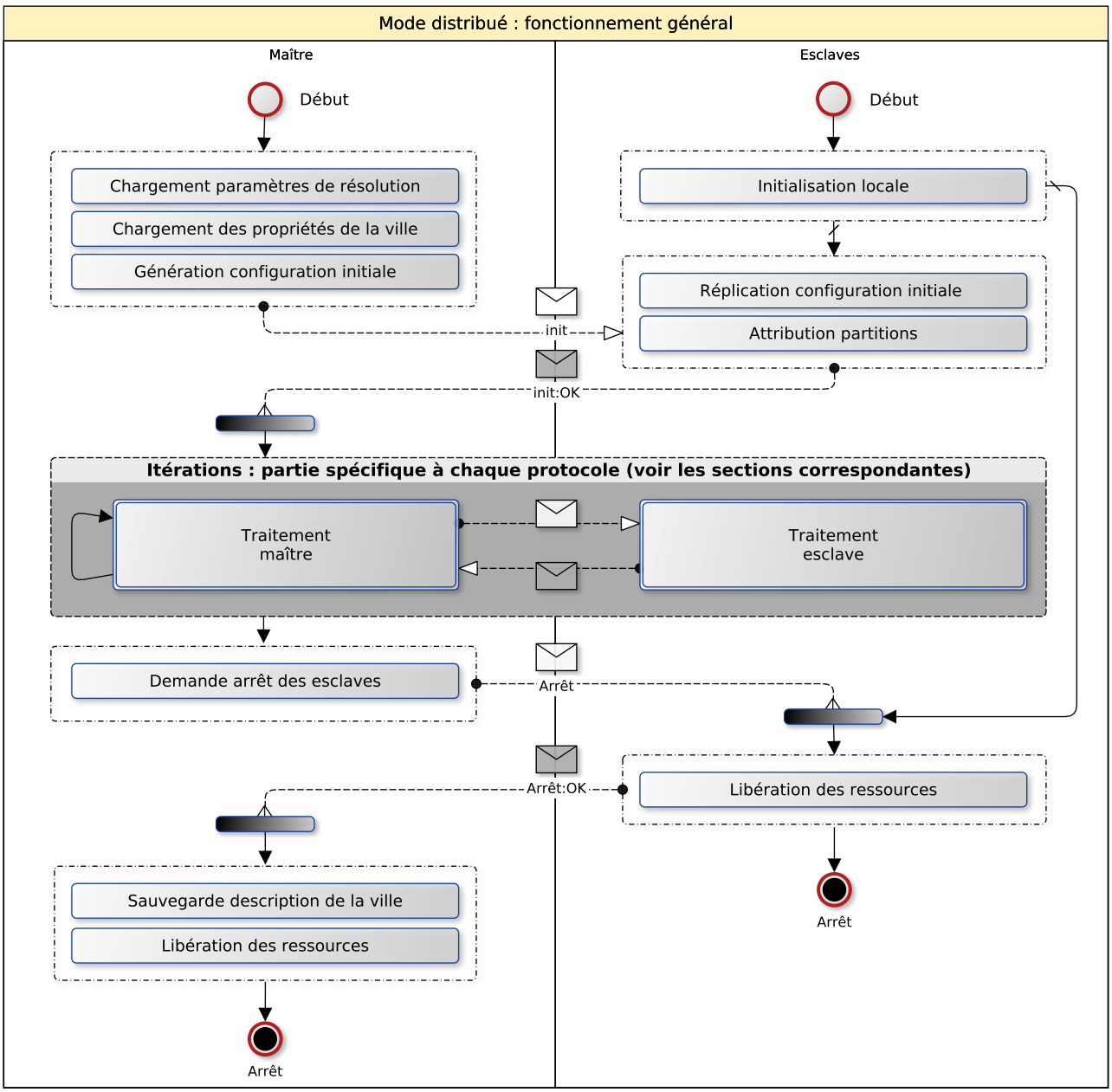
/* Variables de travail :
- $bestPermutations$ est la liste triée (par ordre croissant de variation de coût) des meilleures permutations
- $\triangle_{cost}$ est le gain pouvant être apporté à s en échangeant les valeurs de deux variables
- $worstDeltaCost$ est la variation de coût de la permutation dans $bestPermutations$ la moins bien classée (dernière de la liste) */

$worstDeltaCost \leftarrow 0$;
$bestPermutations \leftarrow \emptyset$;

Fonction EVAL(s, V_i, V_j)
begin
  Calculer $\triangle_{cost}$ en évaluant l'échange des valeurs entre V_i et V_j dans s;
  if *(*$\triangle_{cost} \geq 0$*)* **then**
    return $\triangle_{cost}$;
  if *(*$|bestPermutations| = maxPermutations$*)* **then**
    if *(*$\triangle_{cost} \geq worstDeltaCost$*)* **then**
      return $\triangle_{cost}$;
    else
      Supprimer le dernier élément de $bestPermutations$;
  Création de la permutation p($\triangle_{cost}, V_i, V_j$) ;
  $bestPermutations \leftarrow bestPermutations \cup p$;
  if *(*$|bestPermutations| = maxPermutations$*)* **then**
    $worstDeltaCost \leftarrow$ variation de coût du dernier élément de $bestPermutations$;
  return $\triangle_{cost}$;
end

Fonction APPLY(s)
begin
  forall *(*$p \in bestPermutations$*)* **do**
    Calculer $\triangle_{cost}$ (actualisé) en évaluant l'échange des valeurs entre $p.V_i$ et $p.V_j$ dans s;
    if *(*$\triangle_{cost} \geq 0$*)* **then**
      $valid = false$;
    else
      if *(*$\triangle_{cost} \leq p.\triangle_{cost}$*)* **then**
        $valid = true$;
      else
        $variation = \triangle_{cost} - p.\triangle_{cost}$;
        $valid = (variation \leq \|p.\triangle_{cost} * maxDeviation/100\|)$;
    if *(valid)* **then**
      Échanger les valeurs des variables $p.V_i$ et $p.V_j$ dans s ;
end

FIGURE 9.4 – Mode distribué : fonctionnement général maître-esclaves. La partie « itérations » signalée en foncé est spécifique à chaque protocole distribué que nous proposons et elle fera l'objet d'un schéma complémentaire dans les sections correspondantes.



Au début de l'algorithme (cf. figure 9.4), le processus maître détermine les partitions et transmet les sous-ensembles de variables aux processus esclaves. Il synchronise[6] ensuite le travail de chaque processus esclave, à chaque itération, et collecte les permutations qu'il choisit ou non d'appliquer sur la configuration courante partagée entre tous les processus. Finalement, il détermine les conditions d'arrêt de l'algorithme et retourne la meilleure solution trouvée. Pendant ce temps, chaque processus esclave opère de façon concurrente avec les variables qui lui sont attribuées, de façon à ce que l'ensemble des processus esclaves réunis puisse évaluer à chaque itération la totalité des permutations possibles sur la configuration courante partagée, tout comme le ferait l'algorithme séquentiel.

Avec ce principe, non seulement nous tirons bénéfice de tous les calculs réalisés sans les gaspiller, mais

[6]Dans le sens qu'il attend une réponse (le résultat) de chaque processus esclave avant de poursuivre son travail.

TABLE 9.1 – Principales caractéristiques liées au fonctionnement des différents protocoles distribués.

Caractéristiques	Protocoles			
	1	2	3	4
#phases (séries d'échanges sur le réseau)	2	1	1	1
#$\widehat{P}_E$ par rapport au #partition	#part	#part	#part	#part*(#part+1)/2
évaluation parallèle des meilleurs candidats	O	N	O	O
étendue de recherche des meilleurs candidats	partition	grille	grille	1 à 2 partitions
qui évalue les meilleurs candidats	$\widehat{P}_E$	$\widehat{P}_M$	$\widehat{P}_E$	$\widehat{P}_E$
agrégation des meilleurs candidats dans une liste globale	O	N	N	N
transfert liste des meilleurs candidats	$\widehat{P}_E \rightarrow \widehat{P}_M$ et $\widehat{P}_M \rightarrow \widehat{P}_E$	$\widehat{P}_M \rightarrow \widehat{P}_E$	-	-
transfert du meilleur gain trouvé par candidat ($\widehat{P}_E \rightarrow \widehat{P}_M$)	O	O	O	N
transfert des candidats à bannir ($\widehat{P}_M \rightarrow \widehat{P}_E$)	O	O	O	N
transfert des meilleures permutations trouvées ($\widehat{P}_E \rightarrow \widehat{P}_M$)	O	O	O	O
transfert des permutations appliquées ($\widehat{P}_M \rightarrow \widehat{P}_E$)	O	O	O	O
itération locale si pas de permutation favorable	N	N	N	O
liste globale des candidats bannis présente sur le $\widehat{P}_M$	N	O	N	N
étendue de la liste des candidats bannis sur chaque $\widehat{P}_E$	partition	partition	grille	1 à 2 partitions

nous profitons également de la disposition géographique / géométrique de notre problème pour répartir de façon cohérente les différentes parties de la carte sur les différents processus : un découpage spatial pour un problème spatial.

Dans la suite, nous choisissons de présenter les caractéristiques principales liées au fonctionnement de notre algorithme distribué. En effet, nous proposons différentes implémentations (voir la figure 9.6 qui récapitule le fonctionnement des différents protocoles distribués), et chaque protocole expose des caractéristiques différentes qui permettent de les comparer. Nous étudierons ensuite les éléments complémentaires intégrés au modèle pour supporter le mode distribué. Nous disposerons alors de tous les éléments pour examiner en détail le fonctionnement de chaque protocole.

9.4.1 Caractéristiques liées au fonctionnement de la version distribuée

Chaque processus esclave a en charge une partie des variables de la grille (partition) et dispose d'une représentation synchronisée de la configuration courante globale qui est partagée entre tous les processus, y compris avec le processus maître. Pour un processus esclave, l'enjeu consiste à déterminer localement une liste des meilleures permutations uniquement pour les variables qu'il a en charge, de telle manière que toutes les combinaisons possibles (permutations) puissent être évaluées par la totalité des processus esclaves, de façon analogue au mode séquentiel.

Étant donné que la recherche des meilleures permutations n'affecte pas l'état de la configuration globale partagée entre tous les processus, les processus esclaves peuvent fonctionner en parallèle sans aucune difficulté et sans nécessiter une adaptation conséquente des procédures séquentielles. Pour abréger les écritures, nous utilisons les notations suivantes :

- $\widehat{P}_M$ pour désigner le processus maître ;
- $\widehat{P}_E$ pour désigner un processus esclave ;
- $\widehat{C}_G$ pour désigner la configuration globale qui est partagée et synchronisée entre tous les processus.

Pour illustrer le fonctionnement du mode distribué lié à chaque protocole proposé, nous avons sélectionné quatorze caractéristiques principales (cf. table 9.1) que l'on commente dans les paragraphes suivants.

Nombre de phases
Valeurs possibles :

- 1 : 1 échange réseau lors d'une itération entre le $\widehat{P}_M$ et les $\widehat{P}_E$;
- 2 : 2 échanges réseau lors d'une itération entre le $\widehat{P}_M$ et les $\widehat{P}_E$;

Il s'agit du nombre d'échanges, engagés entre le $\widehat{P}_M$ d'une part et les $\widehat{P}_E$ d'autre part, qui nécessitent une synchronisation. Rappelons qu'il n'y a aucune communication directe entre les $\widehat{P}_E$. En général, une phase commence par un premier message initié par le $\widehat{P}_M$ à destination de tous les $\widehat{P}_E$, se poursuit par les réponses de chaque $\widehat{P}_E$, et prend fin lorsque tous les $\widehat{P}_E$ ont répondu au $\widehat{P}_M$. Une phase regroupe donc plus ou moins de messages qui transitent sur le réseau entre différents processus, chaque message pouvant avoir une taille variable et regrouper des informations de différentes natures. Par exemple, une première phase peut consister à interroger les $\widehat{P}_E$ pour qu'ils fournissent chacun à leur tour une liste locale de leurs meilleurs candidats au $\widehat{P}_M$. Ce message d'interrogation peut contenir les permutations venant d'être réalisées par le $\widehat{P}_M$ et devant être répliquées sur chaque $\widehat{P}_E$ pour synchroniser leur $\widehat{C}_G$. Lorsque le $\widehat{P}_M$ a obtenu toutes les réponses, il peut poursuivre son travail et engager une seconde phase, c'est à dire un second dialogue avec les $\widehat{P}_E$.

Nombre de processus esclaves par rapport au nombre de partitions
Valeurs possibles :

- un entier positif : nombre de $\widehat{P}_E$ (déterminé en fonction du nombre de partitions).

Le nombre de $\widehat{P}_E$ nécessaire pour résoudre une instance donnée de notre problème dépend du nombre de partitions qui a été demandé (par un paramétrage). Les partitions correspondent à des regroupements de cellules libres pour un découpage donné de la grille, chaque partition étant de taille identique (à l'arrondi près). Ces regroupements peuvent être opérés de façon plus ou moins sophistiquée. Dans notre implémentation, un découpage simple est réalisé conformément à l'exemple de la figure 9.5. Par principe, on cherche à définir le plus grand nombre de partitions tout en minimisant le nombre de processus nécessaires pour gérer ces partitions. En effet, un nombre important de partitions permet une forte décomposition des variables de notre problème alors qu'un nombre important de processus nécessite la mobilisation de ressources matérielles importantes (avec un processeur dédié par processus) et des échanges réseaux entre le $\widehat{P}_M$ et chaque $\widehat{P}_E$.

Évaluation parallèle des meilleurs candidats
Valeurs possibles :

- O : plusieurs processus sont impliqués au même moment dans la recherche des meilleurs candidats ;
- N : la recherche des meilleurs candidats est réalisée sur un seul processus.

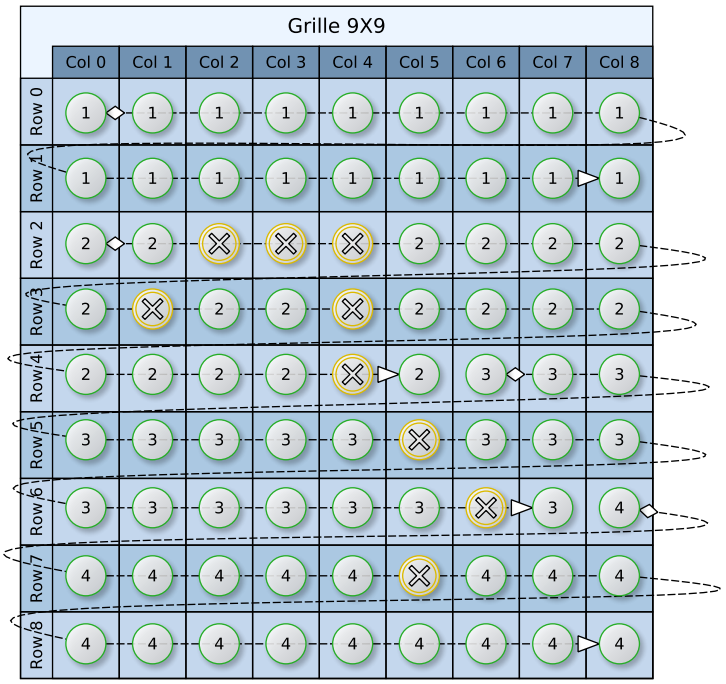
En fonction du protocole, la recherche des meilleurs candidats peut être réalisée soit en bloc par un seul processus, soit être répartie sur différents processus. Cette caractéristique permet de savoir si la recherche est parallélisée.

Étendue de recherche des meilleurs candidats
Valeurs possibles :

- 1 partition : la recherche des meilleurs candidats est effectuée sur une partition au maximum ;
- 1 à 2 partitions : la recherche des meilleurs candidats est effectuée sur 1 ou 2 partitions au maximum ;
- grille : la recherche des meilleurs candidats est effectuée sur la grille entière ;

En fonction du protocole, la recherche des meilleurs candidats peut être réalisée soit à l'échelle de la grille entière, soit à l'échelle d'une seule partition ou bien à l'échelle d'une à deux partitions.

FIGURE 9.5 – Exemple de découpage d'une grille pour former 4 partitions de même taille : chaque groupe est composé de 18 cellules libres, les cellules fixes étant repérées par un caractère « X ».



Qui évalue les meilleurs candidats
Valeurs possibles :

- $\widehat{P}_M$: c'est le $\widehat{P}_M$ qui évalue les meilleurs candidats ;
- $\widehat{P}_E$: chaque $\widehat{P}_E$ évalue les meilleurs candidats.

En fonction du protocole, la recherche des meilleurs candidats peut être prise en charge soit par le $\widehat{P}_M$, soit par chaque $\widehat{P}_E$.

Agrégation des meilleurs candidats dans une liste globale
Valeurs possibles :

- O : une agrégation des meilleurs candidats dans une liste globale est nécessaire ;
- N : aucune agrégation n'est réalisée.

Lorsque la recherche des meilleurs candidats est effectuée à l'échelle d'une partition, la question est de savoir si le protocole nécessite d'agréger les résultats de toutes les recherches à partir de toutes les partitions pour constituer une liste globale des meilleurs candidats.

Transfert liste des meilleurs candidats
Valeurs possibles :

- $\widehat{P}_E \rightarrow \widehat{P}_M$: on transfert la liste des meilleurs candidats de chaque $\widehat{P}_E$ vers le $\widehat{P}_M$;
- $\widehat{P}_M \rightarrow \widehat{P}_E$: on transfert la liste des meilleurs candidats du $\widehat{P}_M$ vers chaque $\widehat{P}_E$;
- - : aucun transfert n'est nécessaire.

Une fois établie, on se demande si la liste des meilleurs candidats doit transiter entre le $\widehat{P}_M$ et les $\widehat{P}_E$. Notons que la liste des meilleurs candidats qui transite peut être globale (i.e. concerner la grille entière) ou être partielle (i.e. limitée à une ou deux partitions).

Transfert du meilleur gain trouvé par candidat des processus esclaves vers le processus maître
Valeurs possibles :

- O : le meilleur gain trouvé par candidat doit être transmis des $\widehat{P}_E$ vers le $\widehat{P}_M$;
- N : aucun transfert n'est nécessaire.

Lorsqu'on ne retrouve pas de permutation, pour un candidat prometteur, dans la liste locale des meilleures permutations trouvées, cela ne signifie pas pour autant qu'il s'agit d'un mauvais candidat. En effet, la liste des meilleures permutations étant limitée en taille, on ne conserve que les meilleures permutations lors d'une itération. Donc, il se peut que d'autres candidats aient de meilleures permutations en nombre suffisant pour ne pas considérer celles du candidat en question. D'autre part, lorsqu'il s'agit d'un candidat global, il peut n'avoir des permutations bénéfiques que sur une seule partition, ce qui est suffisant pour ne pas le bannir. Afin de savoir si un candidat prometteur doit être banni ou non, il faut donc récupérer les meilleurs gains trouvés de chaque candidat sur le $\widehat{P}_M$, ce qui nécessite dans ce cas un transfert de données.

Transfert des candidats à bannir du processus maître vers les processus esclaves
Valeurs possibles :

- O : le $\widehat{P}_M$ doit transmettre aux $\widehat{P}_E$ la liste des candidats à bannir ;
- N : aucun transfert n'est nécessaire.

Il arrive que les processus esclaves seuls n'aient pas une visibilité suffisante pour déterminer si il faut bannir ou non un candidat prometteur. Dans ce cas, l'information doit provenir du $\widehat{P}_M$ qui est en mesure d'agréger les résultats des différents $\widehat{P}_E$ pour en déduire une information pertinente.

Transfert des meilleures permutations trouvées des processus esclaves vers le processus maître
Valeurs possibles :

- O : les $\widehat{P}_E$ doivent transmettre au $\widehat{P}_M$ leur liste des meilleurs permutations ;
- N : aucun transfert n'est nécessaire.

Le but du mode distribué étant de répartir la recherche des meilleures permutations sur les différents $\widehat{P}_E$ puis de récupérer le résultat sur le $\widehat{P}_M$, un transfert réseau est obligatoire pour chaque protocole.

Transfert des permutations appliquées du processus maître vers les processus esclaves
Valeurs possibles :

- O : le $\widehat{P}_M$ doit transmettre aux $\widehat{P}_E$ les meilleures permutations qu'il a retenu et appliqué sur sa $\widehat{C}_G$;
- N : aucun transfert n'est nécessaire.

Les différents processus que nous exploitons peuvent être localisés soit sur la même machine, soit sur des machines distantes. Pour partager la $\widehat{C}_G$ entre ces différents processus, nous n'utilisons pas de mécanisme automatique de mémoire partagée. Pour que la $\widehat{C}_G$ reste synchronisée, il faut donc répliquer systématiquement les modifications effectuées sur chaque processus. Le $\widehat{P}_M$ est le seul habilité à opérer des modifications sur la $\widehat{C}_G$, c'est à dire à appliquer des permutations. Il faut donc acheminer les permutations qu'il a réalisé vers les $\widehat{P}_E$ pour que chaque $\widehat{P}_E$ puisse reproduire ces modifications, c'est à dire rejouer localement chaque permutation sur sa $\widehat{C}_G$. L'ensemble des permutations réalisées à l'occasion d'une itération est reporté en une seule étape du $\widehat{P}_M$ vers les $\widehat{P}_E$.

Itération locale si pas de permutation favorable
Valeurs possibles :

- O : Si aucune permutation favorable n'est trouvée par un $\widehat{P}_E$, un dispositif est prévu pour refaire une nouvelle recherche localement sans en informer le $\widehat{P}_M$;

- N : Si aucune permutation favorable n'est trouvée par un $\widehat{P}_E$, il retourne une liste vide au $\widehat{P}_M$ et se met en attente.

Si un $\widehat{P}_E$ ne trouve pas de permutation favorable après avoir fait une première recherche à partir de la liste des meilleurs candidats, il a deux possibilités : soit il est capable de lancer une nouvelle recherche localement avec de nouveaux critères (renouvellement des candidats) sans communiquer avec le $\widehat{P}_M$, soit il est incapable de relancer une nouvelle recherche. Dans ce cas, il retourne une liste (de ses meilleures permutations) vide et il attend les nouvelles instructions du $\widehat{P}_M$.

Liste globale des candidats bannis présente sur le processus maître
Valeurs possibles :

- O : le $\widehat{P}_M$ doit mémoriser une liste globale des candidats bannis ;
- N : le $\widehat{P}_M$ n'a pas besoin de mémoriser une liste globale des candidats bannis.

En fonction du protocole en œuvre, il peut être nécessaire de maintenir une liste globale des candidats bannis sur le $\widehat{P}_M$.

Étendue de la liste des candidats bannis sur chaque processus esclave
Valeurs possibles :

- 1 partition : la taille de la liste des candidats bannis présente sur chaque $\widehat{P}_E$ est relative à 1 partition ;
- 1 à 2 partitions : la taille de la liste des candidats bannis présente sur chaque $\widehat{P}_E$ est relative à 1 ou 2 partitions ;
- grille : la taille de la liste des candidats bannis présente sur chaque $\widehat{P}_E$ est relative à la grille entière ;

L'étendue de la liste des candidats bannis présente sur chaque $\widehat{P}_E$ varie en fonction du protocole utilisé. Elle peut correspondre à 1 partition, à 1 ou 2 partitions, ou à la totalité de la grille.

9.4.2 Compléments intégrés au modèle

Avant d'identifier les compléments intégrés à notre modèle pour gérer le mode distribué, nous avons besoin de définir la notion de partition.

Definition 12 (Partition)**.** Une partition correspond à un regroupement de cellules libres de la grille, le nombre de partitions étant imposé par un paramétrage et chaque partition étant de même taille (à l'arrondi près). Toutes les cellules libres de la grille sont affectées aux partitions. Pour associer une cellule à une partition, un découpage simple de la grille est réalisé conformément à l'exemple de la figure 9.5.

Compte tenu du regroupement des cellules en partitions, et de la répartition des partitions entre différents $\widehat{P}_E$, nous introduisons les informations suivantes associées à chaque cellule de la grille :

- *numéro de groupe* : pouvant prendre la valeur « 0 », « 1 » ou « 2 » ;
- *indicateur de sélection* : pouvant prendre la valeur « vrai » ou « faux ».

Ces informations sont intégrées dans la $\widehat{C}_G$ de chaque $\widehat{P}_E$ sans pour autant être synchronisées entre les différents processus. Elles sont prévues pour restreindre les permutations pouvant être réalisées sur chaque $\widehat{P}_E$ (cf. algorithme 15).

9.4.2.1 Numéro de groupe

Chaque $\widehat{P}_E$ se charge d'attribuer un numéro de groupe aux cellules libres qui sont sous sa responsabilité, c'est à dire qui appartiennent à ses partitions. Un $\widehat{P}_E$ peut avoir une ou deux partitions qui lui sont associées.

Si une seule partition est attribuée au $\widehat{P}_E$, toutes les cellules ont un numéro de groupe égal à « 0 » ; Par contre, si il a deux partitions sous sa responsabilité, il attribue aux cellules de sa première partition un numéro de groupe à « 1 » et aux cellules de sa seconde partition un numéro de groupe égale à « 2 ».

Algorithme 15 : Contrôle de validité d'une permutation (échange de valeurs entre deux variables).

```
1  /* Paramètre donné en entrée :
     • cellFrom est la cellule d'origine impliquée dans la permutation
     • cellTo est la cellule de destination impliquée dans la permutation
     • checkCellFrom indique si il faut que la cellule d'origine soit obligatoirement sélectionnable ? */
2  Fonction ACCEPTABLE(cellFrom, cellTo, checkCellFrom)
3  begin
4      if ( ! cellFrom.libre) then
5          return false ;
6      if ( ! cellTo.libre) then
7          return false ;
8      if (checkCellFrom et ( ! cellFrom.selectionnable) then
9          return false ;
10     if ( ! cellTo.selectionnable) then
11         return false ;
12     if (cellFrom.formeUrbaine = cellTo.formeUrbaine) then
13         return false ;
14     if (cellFrom.intensite != cellTo.intensite) then
15         return false ;
16     if (cellFrom.groupe = cellTo.groupe) et (cellFrom.groupe != 0) then
17         return false ;
18     return true ;
   end
```

Dès lors, pour qu'un $\widehat{P}_E$ puisse autoriser une permutation entre deux cellules, il faut que les conditions suivantes soient remplies (cf. algorithme 15, lignes 16-17) :

- soit les deux cellules ont un numéro de groupe à zéro : ce qui revient à autoriser les permutations au sein d'une même partition ;
- soit les deux cellules ont un numéro de groupe différent : ce qui limite les permutations aux cellules de partitions différentes.

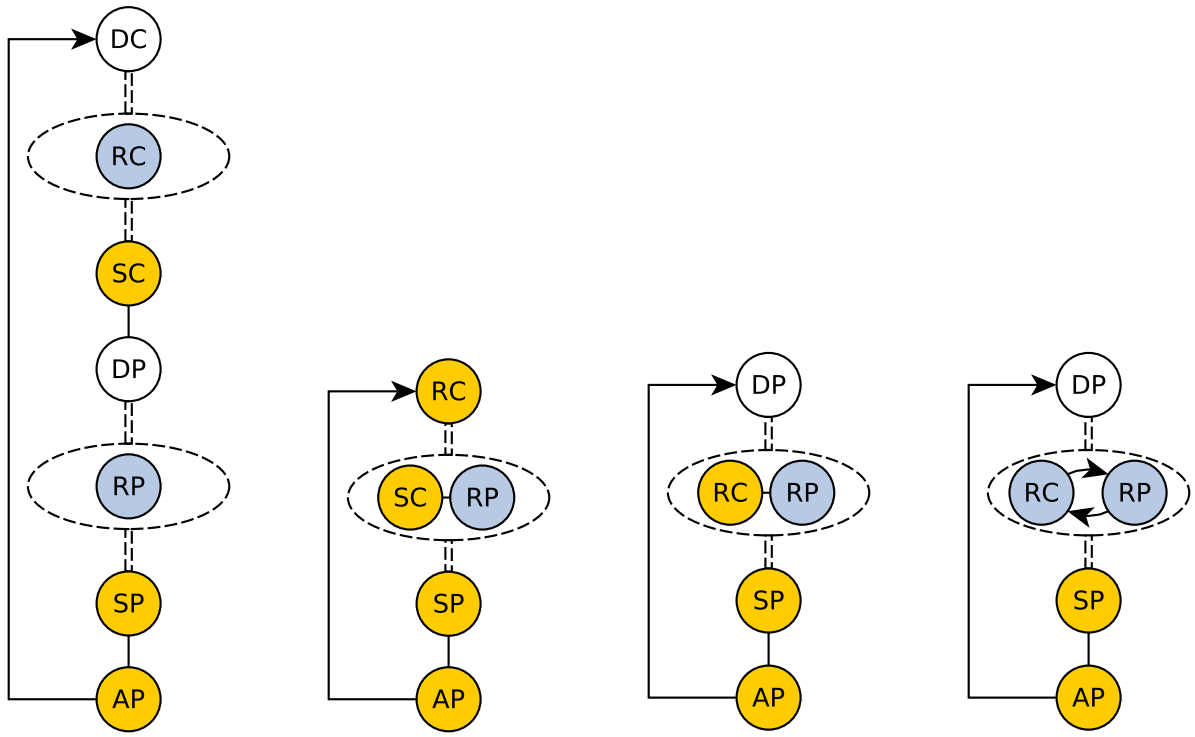
9.4.2.2 Indicateur de sélection

Definition 13 (Indicateur de sélection)**.** Un indicateur de sélection spécifie si une cellule donnée de la grille peut être utilisée pour participer à une permutation. On dira d'une cellule qu'elle est sélectionnable si l'indicateur de sélection qui lui est associé est positionné à vrai.

L'indicateur de sélection est positionné à « vrai » pour toutes les cellules qui appartiennent aux partitions affectées à chaque $\widehat{P}_E$. Une permutation n'est possible qu'entre cellules sélectionnables (cf. algorithme 15, lignes 8-11) sauf lorsque le candidat (cellule à l'origine des permutations) est imposé au $\widehat{P}_E$ par le $\widehat{P}_M$. Dans ce cas, la cellule d'origine liée à la permutation peut correspondre à un candidat provenant des autres partitions.

Nous verrons, dans l'étude de certains protocoles (n° 1 et 4), que cet indicateur de sélection est également utilisé pour limiter la recherche des meilleurs candidats sur chaque $\widehat{P}_E$ aux partitions qui leurs sont associées. Cela correspond à la propriété « *étendue de recherche des meilleurs candidats* » que nous venons de décrire (cf. paragraphe 9.4.1).

FIGURE 9.6 – Comparaison du fonctionnement général de chaque protocole distribué pour une itération. Avec : (DC) Demande Candidats ; (RC) Recherche Candidats ; (SC) Synchronisation Candidats ; (DP) Demande Permutations ; (RP) Recherche Permutations ; (SP) Synchronisation Permutations ; (AP) Application Permutations. Le bleu représente un travail à l'échelle d'une partition et le jaune au niveau de la grille entière. Un ovale en pointillé englobe les tâches réalisées par les esclaves, le reste étant pris en charge par le processus maître. Les doubles traits pointillés indiquent un échange via le réseau.



Protocole 1 Protocole 2 Protocole 3 Protocole 4

9.4.3 Protocoles distribués

Les quatre protocoles distribués que nous proposons (voir figure 9.6) exploitent différemment le découpage de la grille pour identifier à chaque itération les meilleurs candidats puis les meilleures permutations gagnantes qui leur sont associées. Ces approches différentes sont synthétisées dans la table 9.1 qui reprend pour chaque protocole ses principales caractéristiques. En nous appuyant sur ces caractéristiques, nous expliquons maintenant le fonctionnement détaillé de chaque protocole, sachant que la grille globale est toujours découpée en partitions de même taille, le nombre de partitions étant un paramètre de l'application.

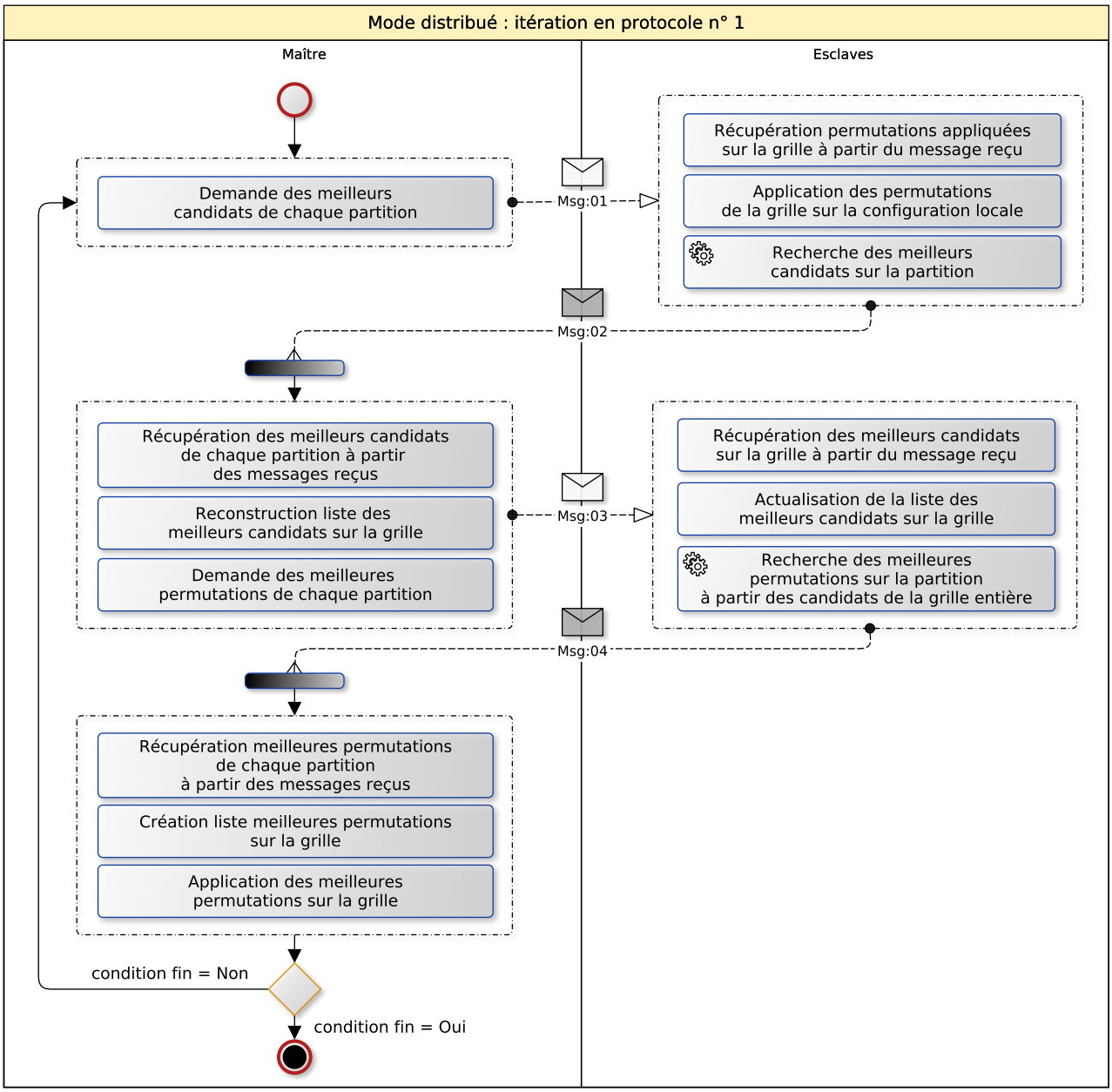
Soulignons dès à présent que nos expérimentations (voir le chapitre 11) sont essentiellement basées sur le protocole n° 1 et le protocole n° 4.

9.4.3.1 Protocole n° 1

Dans ce protocole, chaque $\widehat{P}_E$ est définitivement associé à une partition différente de la grille. À chaque itération, il est nécessaire avec ce protocole de réaliser deux échanges entre le $\widehat{P}_M$ et les $\widehat{P}_E$.

Dans un premier temps, le $\widehat{P}_M$ demande aux $\widehat{P}_E$ d'identifier leurs meilleurs candidats. Chaque $\widehat{P}_E$ recherche alors ses meilleurs candidats parmi les variables qui lui sont attribuées pour la configuration globale partagée (concerne uniquement les cellules de $\widehat{C}_G$ avec un *indicateur de sélection* à vrai). Grâce aux informations collectées sur chaque $\widehat{P}_E$, le $\widehat{P}_M$ peut constituer une liste globale des meilleurs candidats.

FIGURE 9.7 – Mode distribué : traitements réalisés à chaque itération avec le protocole n° 1. Cette partie vient compléter le schéma relatif au fonctionnement général maître-esclaves pour la section « Itérations » (cf. figure 9.4).



Dans un deuxième temps, le $\widehat{P}_M$ demande aux $\widehat{P}_E$ d'identifier leurs meilleures permutations à partir de la liste globale des meilleurs candidats qu'il leur transmet. Chaque $\widehat{P}_E$ recherche alors les meilleures permutations possibles en considérant la liste des meilleurs candidats qui lui est imposée avec les cellules qui lui sont attribuées.

Le $\widehat{P}_M$ collecte les meilleures permutations provenant de chaque $\widehat{P}_E$ et reconstitue une liste globale des meilleures permutations. Pour reconstituer la liste globale des meilleures permutations, le $\widehat{P}_M$ ajoute à cette liste les meilleures permutations provenant de chaque $\widehat{P}_E$ pour ne conserver que les meilleures (i.e. celles qui apportent le meilleur gain) dans la limite d'une taille maximum définie par un paramètre.

Il applique alors ces permutations sur la $\widehat{C}_G$ (comme on le ferait avec la version séquentielle). Il poursuit en demandant à nouveau les meilleurs candidats aux $\widehat{P}_E$, tout en profitant de cet échange (i.e. message) pour transmettre les permutations qu'il a effectivement réalisé afin que chaque $\widehat{P}_E$ puisse synchroniser

localement sa $\widehat{C}_G$.

Par rapport à l'algorithme séquentiel, le fonctionnement diffère dans la mesure où une liste de candidats bannis doit être gérée localement sur chaque $\widehat{P}_E$, la recherche des candidats bannis étant de leur ressort. Dans ces conditions, les dimensions dynamiques de la liste des candidats bannis sont calculées par rapport au nombre de variables libres attribuées à chaque $\widehat{P}_E$. De plus, pour identifier les mauvais candidats qui doivent être insérés dans ces listes locales, il est nécessaire de croiser les informations du $\widehat{P}_M$ et des $\widehat{P}_E$ à chaque itération. Pour ce faire, après avoir identifié localement ses meilleures permutations, chaque $\widehat{P}_E$ transmet au $\widehat{P}_M$ le meilleur gain trouvé par candidat imposé. Ces informations sont agrégées par le $\widehat{P}_M$ qui détermine pour chaque candidat le meilleur gain obtenu. Il retourne ensuite aux $\widehat{P}_E$ la liste globale agrégée des candidats incapables de générer un gain. Charge ensuite à chaque $\widehat{P}_E$ de mémoriser dans sa liste locale des candidats bannis les mauvais candidats reçus qui font partie des cellules dont il a la charge.

Les principes mis en œuvre avec ce protocole sont simples mais ce dernier laisse néanmoins apparaître de nombreuses faiblesses :

- chaque itération nécessite une communication en 2 phases entre le $\widehat{P}_M$ et les $\widehat{P}_E$ pour identifier les meilleurs candidats puis les meilleures permutations ;
- si on travaille avec un nombre réduit de meilleurs candidats et que les candidats considérés n'apportent pas de gain, deux échanges (communications) sont réalisés entre le $\widehat{P}_M$ et les $\widehat{P}_E$ sans améliorer la $\widehat{C}_G$;
- à l'inverse, si on travaille avec beaucoup de candidats, on augmente le nombre de permutations à évaluer séquentiellement sur chaque $\widehat{P}_E$, ce qui pénalise fortement les temps de calcul ;
- comme on recherche les meilleures permutations à partir des mêmes candidats sur chaque $\widehat{P}_E$, l'origine des permutations remontées au $\widehat{P}_M$ est réduite aux mêmes candidats, ce qui limite la diversité spatiale des permutations et l'efficacité du mode multi-permutations.

Pour contourner certaines de ces difficultés, une solution consiste à aménager le mode multi-candidats pour stopper le processus de recherche des meilleures permutations sur chaque $\widehat{P}_E$ dès que le processus a rencontré « x » candidats offrant un gain (« x » étant un paramètre de l'application). Mais dans ce cas, l'identification des candidats bannis devient un peu plus complexe et on s'éloigne un peu plus du fonctionnement séquentiel de l'algorithme. D'autre part, le temps de calcul pour les différents $\widehat{P}_E$ risque alors de varier fortement, la synchronisation entre les différents $\widehat{P}_E$ rendant le temps de chaque itération dépendante du $\widehat{P}_E$ le plus long. Par exemple, imaginons pour une itération donnée que la liste globale des meilleurs candidats imposés soit composée des candidats $\{c_1, c_2, \ldots, c_{10}\}$ et que le processus de recherche des meilleures permutations soit limité aux $x = 2$ premiers candidats offrant des gains. Si $\{c_3, c_5, c_6, c_8, c_{10}\}$ sont les seuls bons candidats de la liste globale pour un $\widehat{P}_E$, ce dernier devra évaluer les permutations uniquement pour $\{c_1, c_2, c_3, c_4, c_5\}$; c_3 et c_5 lui permettant d'honorer l'obligation de deux candidats avec gains.

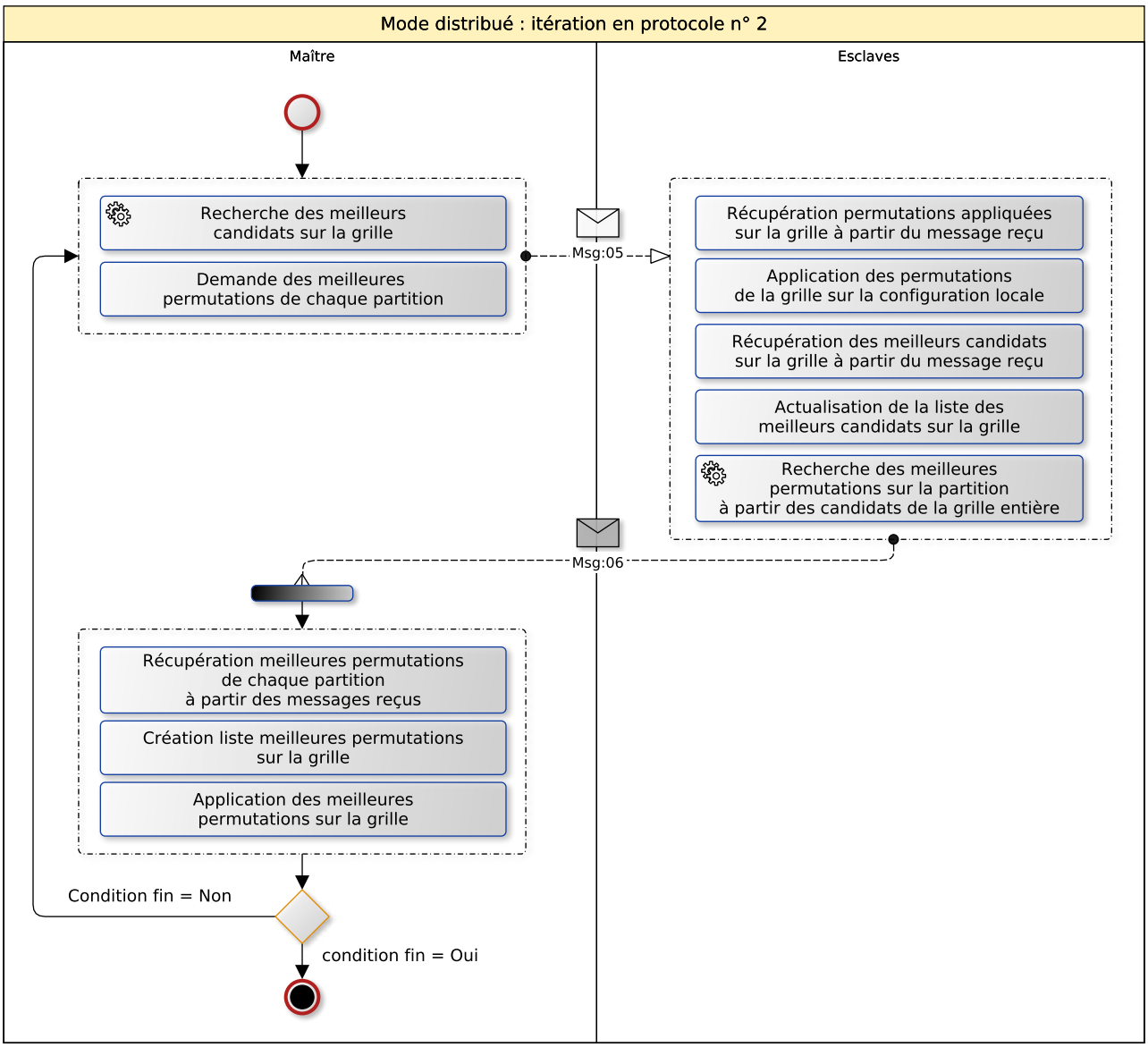
Dans le même esprit, une autre possibilité (celle-ci n'ayant pas été implémentée) consisterait à poursuivre l'évaluation des permutations à partir de la liste des candidats imposés jusqu'à ce qu'un nombre minimum de permutations bénéfiques soit atteint.

9.4.3.2 Protocoles n° 2 et 3

Ces deux protocoles tentent d'améliorer le protocole n° 1 en éliminant une phase de communication sur les deux obligatoires sans toutefois augmenter le nombre de $\widehat{P}_E$ nécessaires. Ils fonctionnent en une seule phase avec un nombre de partitions (lié au découpage de la grille) égal au nombre de $\widehat{P}_E$, chaque $\widehat{P}_E$ étant définitivement associé à une partie différente de la grille. Pour pouvoir fonctionner dans ces conditions, ces protocoles abandonnent l'idée d'une recherche parallèle et fractionnée des meilleurs candidats et font le choix d'identifier à chaque itération les meilleurs candidats de façon globale et séquentielle (par un processus) sur la totalité de la grille.

Spécificités du protocole n° 2 : meilleurs candidats identifiés par le processus maître

FIGURE 9.8 – Mode distribué : traitements réalisés à chaque itération avec le protocole n° 2. Cette partie vient compléter le schéma relatif au fonctionnement général maître-esclaves pour la section « Itérations » (cf. figure 9.4).

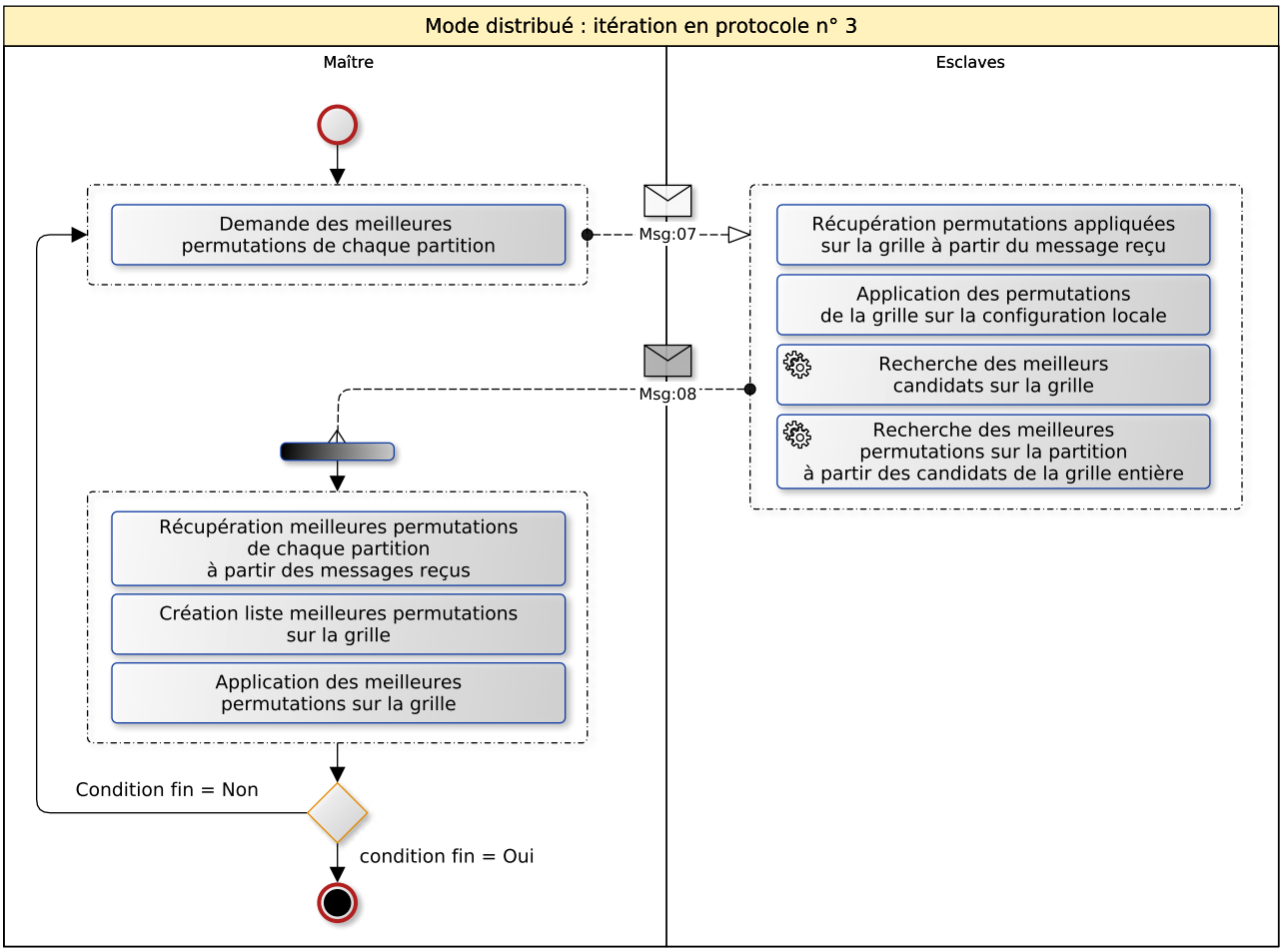


Pour ce deuxième protocole, l'identification des meilleurs candidats est totalement prise en charge par le $\widehat{P}_M$, la liste étant ensuite transmise à tous les $\widehat{P}_E$ en transitant par le message de demande des meilleures permutations. En retour, chaque $\widehat{P}_E$ retourne sa liste des meilleures permutations avec le score des candidats effectivement évalués. Bien que les candidats bannis soient identifiés par le $\widehat{P}_M$, il est utile pour un $\widehat{P}_E$ de connaître les candidats bannis correspondant aux cellules de sa partition pour ne pas avoir à évaluer des permutations y faisant référence. Compte tenu de cette remarque, on doit synchroniser la liste locale des candidats bannis sur chaque $\widehat{P}_E$.

Spécificités du protocole n° 3 : meilleurs candidats identifiés par chaque processus esclave

Pour ce troisième protocole, la charge d'identifier les meilleurs candidats sur la grille complète revient à chaque $\widehat{P}_E$. À partir de cette liste générée à l'identique par chaque $\widehat{P}_E$, les $\widehat{P}_E$ recherchent les meilleures permutations gagnantes générées entre chaque candidat et les cellules sous leur responsabilité. Comme les meilleurs candidats sont recherchés par chaque $\widehat{P}_E$ sur l'ensemble de la grille, une liste des candidats bannis

FIGURE 9.9 – Mode distribué : traitements réalisés à chaque itération avec le protocole nº 3. Cette partie vient compléter le schéma relatif au fonctionnement général maître-esclaves pour la section « Itérations » (cf. figure 9.4).



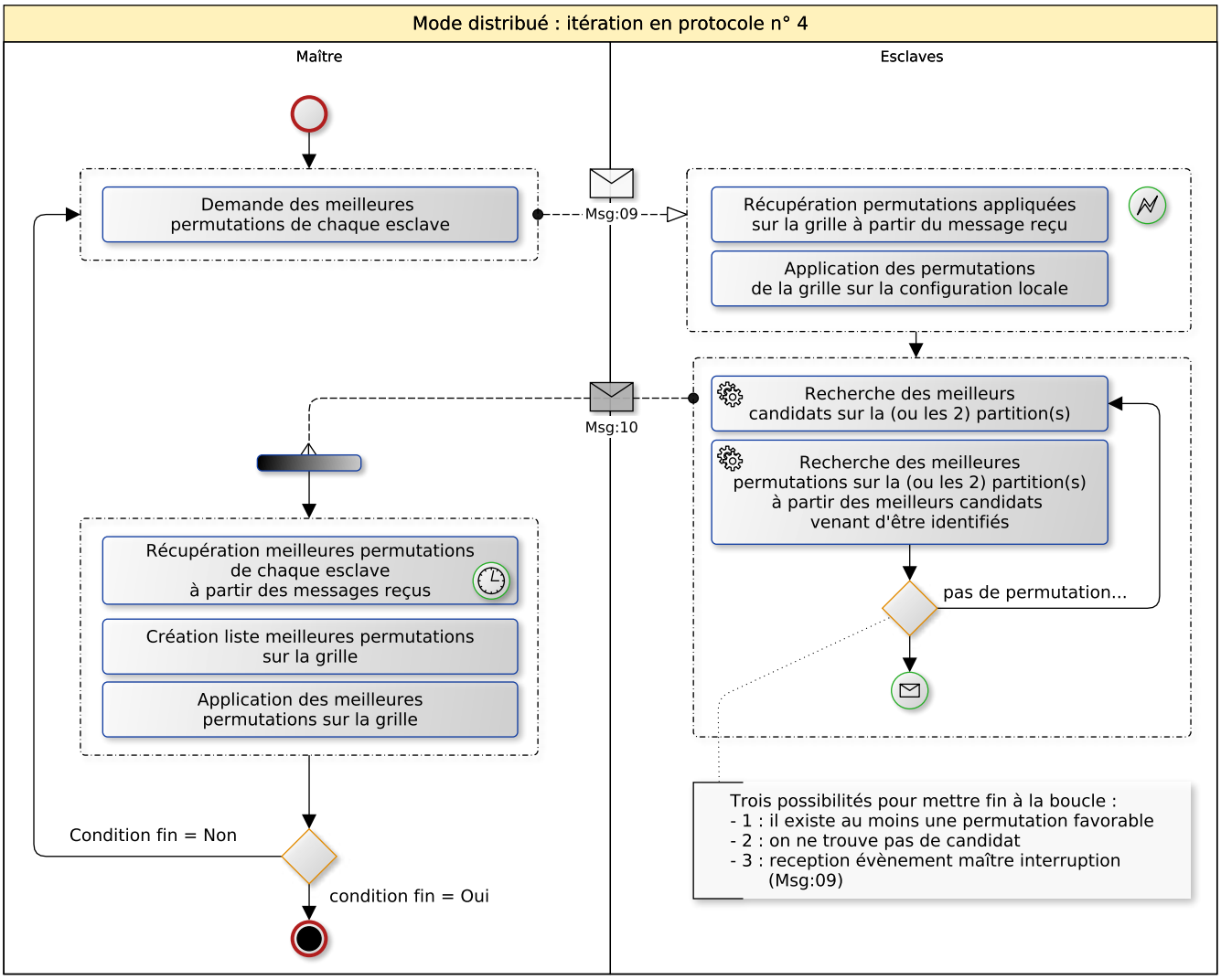
relative à la totalité de la grille doit être synchronisée sur chaque $\widehat{P}_E$ et collaborant avec le $\widehat{P}_M$.

Ces deux protocoles éliminent une phase de communication (avec synchronisation) par rapport au protocole nº 1 tout en exploitant le même nombre de $\widehat{P}_E$. En contrepartie, l'évaluation séquentielle requise pour identifier la liste globale des meilleurs candidats peut être pénalisante. D'autre part, même si le nombre de phases a été réduit, la quantité d'informations devant transiter entre le $\widehat{P}_M$ et les $\widehat{P}_E$ reste quasiment la même (avec moins de messages, mais plus d'informations dans chaque message), les $\widehat{P}_E$ devant faire appel au $\widehat{P}_M$ pour synchroniser leur liste locale des candidats bannis. Finalement, les difficultés principales du protocole nº 1 relatives à la diversité spatiale des permutations et à la perte de temps occasionnée si aucun candidat prometteur ne produit une permutation gagnante restent présentes.

9.4.3.3 Protocole nº 4

Avec ce protocole, une seule phase (échange réseau synchronisé) est nécessaire à chaque itération entre le $\widehat{P}_M$ et les $\widehat{P}_E$. Néanmoins, pour le même nombre de partitions, il faut prévoir beaucoup plus de $\widehat{P}_E$, avec $|\widehat{P}_E| = |partition| * (|partition| + 1)/2$. Par exemple, pour un grille globale découpée en quatre parties de même taille, il faut prévoir dix $\widehat{P}_E$, chaque processus esclave étant définitivement associé à une ou deux parties de la grille (voir table 9.2).

FIGURE 9.10 – Mode distribué : traitements réalisés à chaque itération avec le protocole nº 4. Cette partie vient compléter le schéma relatif au fonctionnement général maître-esclaves pour la section « Itérations » (cf. figure 9.4).



À chaque itération, chaque $\widehat{P}_E$ recherche les meilleures permutations possibles uniquement entre les parties qui lui sont assignées. Par exemple, le $\widehat{P}_E$ nº 1 de la table 9.2 examine les permutations des cellules localisées dans la zone ① avec les autres cellules de la zone ①. Le $\widehat{P}_E$ nº 5 examine les permutations des cellules de la zone ① avec les cellules de la zone ②, etc. Cette façon de procéder permet de couvrir localement l'ensemble des permutations possibles sur la grille globale.

La recherche des meilleures permutations est bien entendu soumise à une liste des meilleurs candidats établie préalablement. Lors d'une itération, chaque $\widehat{P}_E$ commence par constituer en fonction des parties qui lui sont confiées sa propre liste locale des meilleurs candidats. Pour un $\widehat{P}_E$ affecté à une seule partie, il recherche ses meilleurs candidats uniquement sur cette partie. Pour un $\widehat{P}_E$ affecté à deux parties, il recherche ses meilleurs candidats sur les deux parties (suivant l'*indicateur de sélection* décrit en section 9.4.2.2). Pour chaque $\widehat{P}_E$, nous disposons d'une liste locale des candidats bannis dont la taille dynamique dépend du nombre de cellules libres associées au $\widehat{P}_E$. Chaque candidat, de la liste locale des meilleurs candidats, incapable de fournir le moindre bénéfice est directement ajouté à la liste locale des candidats bannis.

Les $\widehat{P}_E$ sont synchronisés avec le $\widehat{P}_M$ qui collecte les meilleures permutations et décide de celles qui devront être appliquées. Une fois cette opération réalisée, le $\widehat{P}_M$ demande à nouveau les meilleures per-

TABLE 9.2 – Protocole n° 4 : exemple de décomposition de la carte d'une ville en quatre parties de taille uniforme (à gauche) et distribution de ces parties sur dix processus esclaves (à droite).

①	②
③	④

Esclave	1	2	3	4	5	6	7	8	9	10
Partie	①	②	③	④	①;②	①;③	①;④	②;③	②;④	③;④

mutations aux $\widehat{P}_E$ en profitant de cet échange (message) pour transmettre à chaque $\widehat{P}_E$ les permutations qu'il a effectivement réalisé sur la $\widehat{C}_G$. Une fois synchronisés, les $\widehat{P}_E$ peuvent rechercher les nouvelles permutations en parallèle pour la même $\widehat{C}_G$.

Précisons tout de même que lorsqu'un $\widehat{P}_E$ ne trouve aucune permutation favorable à partir de sa liste locale des meilleurs candidats, il ne se synchronise pas immédiatement avec le $\widehat{P}_M$. Dans ce cas, au lieu de rendre une réponse sans permutation au $\widehat{P}_M$, il relance automatiquement le processus local de recherche de ses meilleurs candidats avant de rechercher les meilleures permutations correspondantes limitées à ses zones de responsabilité. Le $\widehat{P}_E$ poursuit cette succession de recherches jusqu'à ce que l'un des critères suivants soit atteint :

1. il trouve des permutations favorables ;
2. il ne trouve plus de candidats (sa liste de candidats bannis est pleine et elle ne peut plus être agrandie) ;
3. son $\widehat{P}_M$ le force à synchroniser sa $\widehat{C}_G$.

Concernant le critère n° 3 évoqué ci-dessus, le $\widehat{P}_M$ peut en effet reprendre la main au bout d'un certain temps d'attente : lorsque certains $\widehat{P}_E$ ne fournissent pas de réponse au bout d'un délai (timeout), le $\widehat{P}_M$ abandonne la communication en cours avec les $\widehat{P}_E$ qui n'ont pas été en mesure de répondre dans les délais à sa requête (pour fournir leur liste des meilleures permutations locales). Le $\widehat{P}_M$ poursuit son travail avec les résultats partiels obtenus (application des meilleures permutations) et réinterroge l'ensemble des $\widehat{P}_E$. C'est à ce moment là que les $\widehat{P}_E$ n'ayant pas encore répondus sont obligés de synchroniser leur $\widehat{C}_G$ en abandonnant leur recherche en cours.

Un mécanisme spéciale est mis en œuvre pour écarter les messages obsolètes qui ont pu être émis par les $\widehat{P}_E$ retardataires entre le moment où le $\widehat{P}_M$ a abandonné son attente et le moment où les $\widehat{P}_E$ ont été sommés de se synchroniser. Pour écarter ces messages obsolètes qui peuvent arriver lors des synchronisations ultérieures, un numéro d'itération est intégré dans tous les messages (doit être identique entre une demande et les réponses correspondantes) et les réponses associées à un numéro d'itération inférieur au numéro d'itération courant sont systématiquement ignorées par le $\widehat{P}_M$.

Concernant le délai d'attente (timeout) exploité par le $\widehat{P}_M$, il est difficile de le fixer de façon statique compte tenu du contexte matériel, de la performance des réseaux en place ou encore de l'avancement dans le processus de résolution. Nous avons donc fait le choix de le calculer de façon dynamique à chaque itération. Pour le calculer, le $\widehat{P}_M$ doit attendre la première réponse valide du $\widehat{P}_E$ le plus rapide. Il peut alors définir le timeout lié à l'itération en fonction du temps mis par le $\widehat{P}_E$ le plus rapide, ce temps étant multiplié par un coefficient paramétrable.

9.5 Conclusion

Ce chapitre dresse un inventaire des techniques principales que nous proposons pour générer une pré-programmation urbaine à partir du modèle de contrainte défini pour notre problème d'aménagement urbain.

Dans un premier temps, cet inventaire décrit les méthodes mises au point pour générer des configurations initiales ainsi que notre algorithme de recherche locale basé sur *Adaptive Search*. Après avoir identifié différentes optimisations qui permettent d'améliorer les performances de cette métaheuristique, nous introduisons les modes multi-candidats et multi-permutations qui viennent enrichir les possibilités déjà disponibles. Dans un contexte séquentiel, l'exploitation de plusieurs candidats renforce la recherche d'une meilleure permutation à chaque itération mais au prix d'un effort de calcul supplémentaire puisqu'il faut

évaluer les permutations de chaque candidat. D'un autre côté, le mode multi-permutations profite d'une exploration du voisinage de toute façon nécessaire pour mémoriser les meilleures permutations qui pourront être appliquées en série sans nécessiter d'efforts significatifs de calcul. Ces deux modes peuvent être activés séparément ou fonctionner simultanément. Lorsqu'ils collaborent, le mode multi-candidats peut apporter de la diversité dans les meilleures permutations retenues. Provenant de secteurs géographiques différents, les permutations sélectionnées ont alors plus de chance de ne pas interférer les unes avec les autres, augmentant ainsi le nombre de permutations restant valides pour une application consécutive (i.e. en mode multi-permutations, les permutations retenues lors d'une même itération sont appliquées en cascade).

Pour faire face à des instances de grande taille tout en contenant les temps de résolution, nous optons pour une version distribuée de notre algorithme basée sur une décomposition de variables et pouvant être déployée à grande échelle sur une grille de calcul. Cette version permet alors d'exploiter toute la puissance des modes multi-candidats et multi-permutations sans nécessiter des temps de calcul supplémentaires, différentes portions de la grille étant confiées à des processus œuvrant en parallèle pour identifier à chaque itération les meilleures permutations d'une configuration globale synchronisée. Pour ce mode distribué, nous avons exploré quatre protocoles pouvant être mis en œuvre, le quatrième étant celui qui offre le plus d'autonomie aux processus esclaves qui s'affranchissent dans certaines circonstances des ordres donnés par un processus maître qui sait mieux déléguer ses tâches tout en orchestrant un fonctionnement global de l'algorithme cohérent et efficace.

10

Mode interactif

La seconde étape de notre processus de résolution est interactive. Le but est de permettre aux utilisateurs de garder le contrôle sur la solution à travers des manipulations interactives sur la ville. Dans le sens d'intégrer intelligemment les interactions des utilisateurs, l'idée consiste à maintenir actif le processus de résolution pendant que l'utilisateur effectue des modifications sur la solution.

Lorsque l'on évoque une possible interaction, on sous-entend généralement l'engagement d'au moins deux acteurs (ou composants) impliqués dans une communication pour participer à une activité partagée. D'ailleurs, le dictionnaire LAROUSSE donne au mot *interaction* la définition suivante : « *Action réciproque qu'exercent l'un sur l'autre deux ou plusieurs systèmes physiques* ». Dans notre cas, on comprend aisément l'interaction qui se joue entre les utilisateurs du système et l'interface graphique. Mais de notre point de vue, l'interaction ne s'arrête pas là et elle intègre également le solveur qui doit, à tout moment, réagir aux actions des utilisateurs [149, 171, 254, 255].

Dans ce chapitre, nous revenons sur les grands principes qui ont guidé nos choix pour mettre au point ce mode interactif et nous présentons les différents composants impliqués dans l'interaction. Parmi les parties prenantes, le système qui gère les communications entre les acteurs est considéré comme un élément à part entière contribuant pour une large part à la réussite du processus global.

En contexte interactif, le support de multiples scénarios en lien avec un projet est essentiel du point de vue de l'utilisateur, chaque scénario lui permettant de représenter une solution possible avec ses avantages et inconvénients [249]. Nous commençons donc par présenter nos travaux dans ce domaine.

Nous expliquons ensuite le fonctionnement général de notre système en commençant par détailler les différents paramétrages qui ont un effet indirect sur son comportement interactif. Nous décrivons ensuite les manipulations directes qui peuvent être opérées sur la grille en relevant les principales difficultés techniques qu'elles soulèvent. Enfin, un paragraphe entier est dédié aux composants graphiques mis au point pour informer à la volée les utilisateurs des conséquences de leurs choix.

Ce qui nous importe dans ce chapitre, ce sont les fonctionnalités interactives ; l'aspect esthétique de l'IHM n'étant ici qu'un aspect secondaire. Pour une question de simplification, nous proposons une version 2D de l'IHM qui peut très facilement être transposée en 3D compte tenu de la stratégie qui a guidé nos choix.

10.1 Principes retenus

Pour que l'application soit réactive en toutes circonstances, les traitements gourmands (i.e. avec des temps de calcul supérieurs à la seconde) exécutés à la demande du dispositif tactile doivent être lancés en mode

asynchrone dès que c'est possible : ceci permet aux utilisateurs de poursuivre leurs manipulations sans attendre le résultat des calculs. En contrepartie, une technique (à base de synchronisation[1]) doit être proposée pour intégrer avec un temps de décalage le résultat des calculs asynchrones sans interférer avec les opérations courantes des utilisateurs. De plus, les dispositifs utilisés pouvant être multi-utilisateurs (e.g. grandes tables tactiles multi-touch), il est important que le système proposé puisse supporter le travail concurrent de plusieurs utilisateurs sur des portions différentes d'une même ville.

D'autre part, les contraintes techniques ou bien les préoccupations n'étant pas les mêmes entre le développement d'une IHM et le développement d'un solveur, une exécution autonome et parallèle dans des processus distincts est souhaitée. De cette manière, l'IHM peut être développée dans un langage de programmation ou/et avec des outils différents de ceux retenus pour implémenter le solveur. Finalement, l'IHM et le solveur doivent pouvoir fonctionner de part et d'autre sur des machines distantes, sans que la communication entre les deux processus ne nécessite un couplage fort. Par exemple, on peut imaginer que la partie IHM s'exécute sur l'ordinateur associé physiquement au dispositif tactile tandis que le solveur est déployé sur une grille de calcul répartie sur plusieurs sites [243].

10.2 Présentation des composants impliqués dans l'interaction

Compte tenu des principes annoncés, nous proposons d'organiser l'interaction entre trois composants majeurs (cf. figure 10.1) pour permettre à notre système d'être efficace (i.e. réactif aux demandes des utilisateurs) :

- un gestionnaire de messages (asynchrone) ;
- une application liée au dispositif tactile ;
- un solveur interactif.

10.2.1 Gestionnaire de messages

Notre gestionnaire de messages (GDM) se charge de router les messages d'un processus émetteur à un processus récepteur de façon asynchrone. Il peut être installé sur la machine qui héberge les processus qui vont l'utiliser (émetteur et récepteur) ou bien sur une machine distante. Il n'y a aucune communication directe entre l'émetteur et le récepteur : toutes les communications transitent par le GDM. En mode asynchrone, l'émetteur confie son message au GDM et passe immédiatement à ses autres occupations. En complément du GDM, un gestionnaire d'événements doit être activé sur chaque processus connecté au GDM pour qu'il soit alerté dès qu'un message (lui étant destiné) est disponible.

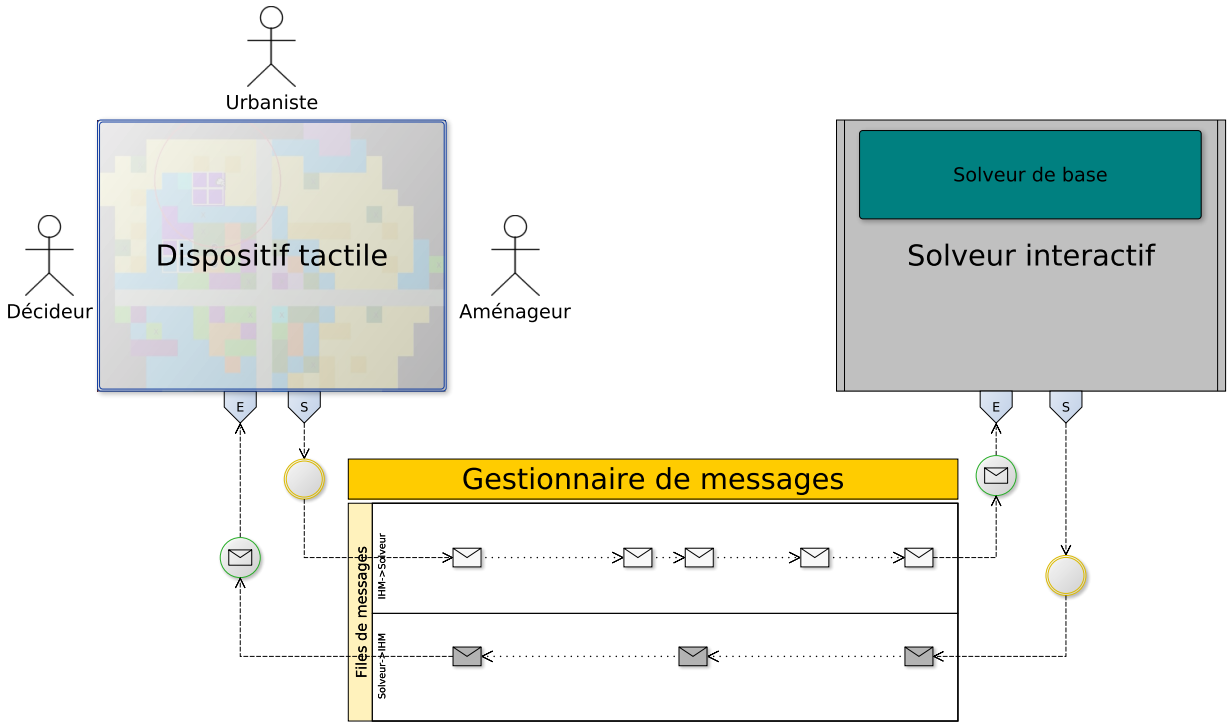
Files de messages Une file de messages conserve les messages qui lui sont confiés tant qu'ils ne sont pas explicitement récupérés par le destinataire. Les messages sont mis en file les uns à la suite des autres sachant que les premiers entrés dans la file sont les premiers sortis. Un nom peut être donné à une file et plusieurs files peuvent cohabiter en même temps. On peut choisir la file à utiliser en fonction de son nom.

Connecteurs d'entrée et de sortie Un connecteur d'entrée et un connecteur de sortie sont installés sur chaque processus voulant travailler avec le GDM. Le connecteur d'entrée peut être vu comme un thread[2] spécialisé capable de détecter à tout moment l'arrivée d'un message sur une file particulière. Dans ce cas, il peut récupérer le message et lancer un traitement particulier. Ensuite, soit il passe au message suivant présent dans la file, soit il se met en attente d'un prochain message à venir.

[1] Dans le sens d'une protection des données partagées entre différents processus légers.

[2] Processus léger qui partage sa mémoire virtuelle et qui est capable d'interagir directement avec le processus principal ou avec d'autres threads associés au processus principal.

FIGURE 10.1 – Routage des messages et interception des événements entre le dispositif tactile et le solveur interactif. Le gestionnaire de messages prend en charge le transfert des informations sur le réseau et permet un couplage faible entre les deux processus. Deux connecteurs repérés par les lettres « E » et « S » permettent à chaque processus (émetteur/récepteur) d'intercepter les messages qui lui sont destinés ou d'injecter de nouveaux messages vers un processus destinataire dans une file donnée. Le solveur interactif incorpore les fonctionnalités du solveur de base mis au point pour la résolution initiale et lui apporte les fonctionnalités nécessaires pour gérer l'interaction.



Le connecteur de sortie permet de créer un nouveau message et de l'injecter dans le GDM sur une file donnée. Les connecteurs d'entrée et de sortie peuvent œuvrer simultanément (en parallèle). Dans ce cas, les traitements associés doivent être prévus pour protéger les données partagées susceptibles d'être modifiées et exploitées par les traitements exécutés en entrée ou en sortie (protection nécessaire via un dispositif de synchronisation).

10.2.2 Dispositif tactile

Le processus interactif doit permettre à un ou plusieurs utilisateurs de modifier l'organisation de la ville en changeant l'emplacement des formes urbaines affectées aux cellules de la grille. On veut conserver un fonctionnement par permutation au niveau de chaque action utilisateur : si l'utilisateur prend une forme urbaine pour la positionner sur une autre cellule, il faudra donc procéder par échange des formes urbaines entre ces deux cellules. Ces permutations imposées par l'opérateur sont alors transmises au solveur interactif pour qu'il prenne en compte ces changements et qu'il calcule une nouvelle solution optimisée. Après cette étape d'optimisation, le dispositif tactile doit être en mesure de récupérer les permutations automatiques opérées par le solveur pour les reporter sur l'IHM.

Dans ce contexte, il faut donc distinguer deux origines possibles pour une permutation :

- soit il s'agit d'une permutation manuelle imposée par l'utilisateur ;
- soit il s'agit d'une permutation automatique issue du processus de résolution pour optimiser la solution courante.

Les permutations manuelles sont issues de l'IHM tandis que les permutations automatiques sont géné-

rées par le solveur interactif. Pendant que le solveur interactif intègre les permutations manuelles et recalcule une nouvelle solution proche de la précédente, il faut néanmoins permettre aux utilisateurs de poursuivre leurs actions sans attendre les résultats des résolutions demandées.

10.2.2.1 Déplacement des formes urbaines

Du point de vue de l'utilisateur, nous représentons le déplacement des formes urbaines comme un « *drag & drop* » qui lui permet de :

- sélectionner une ou plusieurs formes urbaines associées à des cellules d'origine ;
- déplacer le groupe des formes urbaines sélectionnées en faisant apparaître visuellement le groupe et la trajectoire empruntée ;
- lâcher le groupe des formes urbaines sur les emplacements de destination.

Bien qu'il s'agisse d'un procédé classique dans l'utilisation et la perception, sa mise en œuvre relève d'une implémentation particulière qui doit tenir compte de la contrainte de correspondance d'intensité ou des cellules figées (routes, rivières, ...) qui ne doivent pas être modifiées et qui pourront être enjambées au cours du déplacement. On doit également représenter la trajectoire empruntée en indiquant à l'utilisateur si les cellules situées au dessous du groupe en mouvement sont compatibles pour accepter les formes urbaines sélectionnées.

10.2.2.2 Réception et émission de messages

Les actions déterminantes des utilisateurs sont interceptées localement par l'IHM, un message correspondant est généré et confié au GDM à destination du solveur interactif. Chaque message en provenance du GDM et destiné au dispositif tactile est intercepté, interprété et une action adaptée est exécutée localement. C'est toujours le dispositif tactile qui prend le premier l'initiative d'envoyer un message au solveur interactif, ce dernier se mettant au service de l'utilisateur pour exécuter les actions qui lui sont demandées et émettre un message en réponse à la demande lorsque l'action a été réalisée.

10.2.2.3 Liste des Permutations Imposées (LPI)

Lorsque l'opérateur réalise un déplacement de formes urbaines par *drag & drop*, le système génère un nombre de permutations correspondant au nombre de formes urbaines impliquées dans le déplacement. Ces permutations forment alors une liste de permutations imposées (LPI) qui est incorporée dans le message à destination du solveur interactif.

10.2.3 Solveur interactif

Nous voulions utiliser un même solveur que cela soit dans le cadre d'une résolution initiale ou dans le cadre de l'interaction avec les utilisateurs. Pour que cela soit possible, nous encapsulons (enveloppons) le solveur de base (i.e. celui mis au point pour la résolution initiale) afin de le doter de fonctionnalités complémentaires nécessaires à l'interaction, sans pour autant modifier le solveur de base. Le solveur interactif est connecté au dispositif tactile par l'intermédiaire du GDM. Il doit être lancé avant les premières interactions et il se termine automatiquement lorsque l'utilisateur ferme l'IHM. Le solveur interactif charge et mémorise la solution courante dès le lancement de l'application. Il se synchronise (dans le sens d'actualiser) alors avec le dispositif tactile en lui transmettant sa solution courante. Ensuite, à chaque fois qu'une action utilisateur est interceptée par l'IHM, un message est transmis au solveur interactif pour qu'il reporte les opérations manuelles issues de l'IHM sur sa solution courante. Il demande alors au *solveur de base* (celui qu'il encapsule) d'exécuter une résolution initiale basée sur cette solution courante. Le résultat de la résolution calculé par le *solveur de base* devient alors la solution courante du solveur interactif. Cette nouvelle solution est alors communiquée au dispositif tactile pour une nouvelle actualisation des données liées à l'IHM.

Au niveau de notre modélisation, nous consentons quelques compléments sur le modèle (lié au problème d'aménagement urbain) pour mémoriser les choix des utilisateurs en terme de placement des formes urbaines.

10.2.3.1 Modèle : mémorisation des cellules verrouillées par les utilisateurs

En complément des propriétés déjà présentes dans notre modèle (voir chapitre 8), nous proposons d'y intégrer un indicateur (prenant la valeur vrai ou faux) associé à chaque cellule pour indiquer si la forme urbaine affectée à la cellule est verrouillée (i.e. imposée) par l'utilisateur. Tant qu'une forme urbaine est verrouillée à une cellule, le solveur ne peut pas la déplacer de lui même pour optimiser la solution courante.

10.2.3.2 Réception et émission de messages

Un gestionnaire d'événements écoute en permanence les messages en provenance du GDM et destinés au solveur interactif. Dès qu'un message est disponible, il est récupéré, interprété et une action adaptée est exécutée. L'action entreprise par le solveur doit être terminée avant qu'un autre message puisse être « *consommé* » dans la file des messages entrants. En effet, nous souhaitons réutiliser telles quelles les fonctionnalités du solveur de base sans devoir synchroniser (dans le sens de protéger les informations partagées entre les processus légers) ses traitements/données internes. D'autre part, nous avons fait le choix d'empiler les actions des utilisateurs pour les traiter séquentiellement. Cette façon de procéder permet de gérer de façon transparente les actions provenant de plusieurs utilisateurs travaillant simultanément sur des parties différentes d'une même ville. En réponse à l'action entreprise, le solveur peut émettre à son tour un message destiné au dispositif tactile qu'il confie au GDM.

Si les utilisateurs sont très actifs ou/et que les opérations qu'ils demandent au solveur sont gourmandes en temps, les messages en provenance du dispositif tactile vont pouvoir s'empiler dans le GDM et seront traités un à un dès que le solveur sera à nouveau disponible.

10.2.3.3 Liste des permutations imposées (LPI)

Certains messages contiennent des permutations imposées par une action de déplacement (ou plusieurs actions regroupées, ceci étant dû au mode de fonctionnement autonome, cf. section 10.4.3) réalisée par un utilisateur. Dans ce cas, chaque permutation imposée par l'utilisateur est appliquée sur la solution courante en échangeant les formes urbaines des cellules concernées. Pour chaque permutation de la liste prise une à une et dans l'ordre d'arrivée, la cellule d'origine est marquée « déverrouillée » alors que la cellule de destination est marquée « verrouillée ».

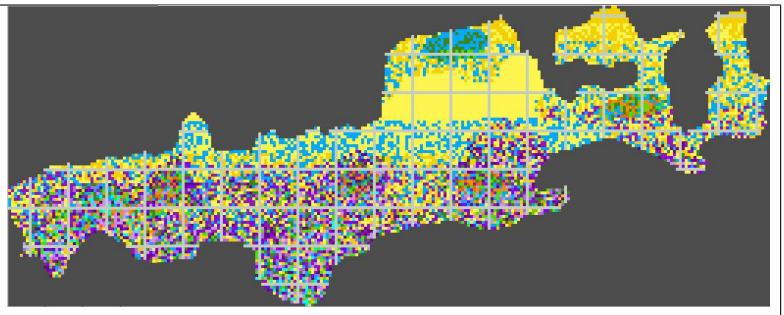
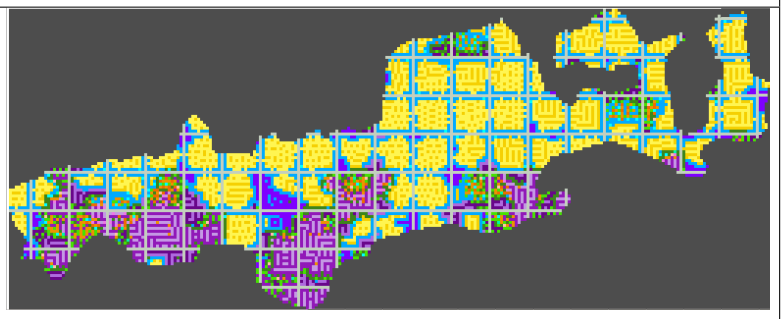
10.2.3.4 Résolution locale

En réponse à certaines actions d'un utilisateur, une résolution peut être lancée pour tenter d'améliorer la solution courante tout en conservant les choix antérieurs réalisés par les opérateurs. Pour chaque action associée à un déplacement de formes urbaines, les permutations imposées par l'utilisateur permettent d'établir un périmètre autour des cellules d'origine et de destination, la taille de ce périmètre étant paramétrable. Avant la résolution, toutes les cellules de la grille sont rendues non sélectionnables sauf celles comprises dans le périmètre identifié. Ensuite, une résolution est demandée et le résultat (formes urbaines qui occupent les cellules) est intégré dans un message à destination du dispositif tactile.

10.3 Gestion des scénarios

Compte tenu du caractère stochastique lié à la procédure de résolution initiale, il est possible de lancer plusieurs fois la procédure à partir des mêmes critères (paramètres, territoire, formes urbaines à répartir

FIGURE 10.2 – Chargement d'un scénario mémorisé préalablement. Dans un premier temps, l'utilisateur indique la ville de référence. Ensuite, il a la possibilité de charger un scénario de travail particulier relatif à cette ville.

MLV avant chargement scénario	
MLV après chargement scénario	

sur la ville, etc.) pour obtenir des solutions différentes, chaque solution étant mémorisée dans une structure XML[3] autonome. À partir de ces différentes solutions, les experts peuvent en sélectionner une qu'ils vont retravailler en opérant diverses modifications. Il faut pouvoir conserver le travail réalisé pour revenir dessus par la suite afin de le compléter, pour le partager avec d'autres personnes ou tout simplement pour en garder une trace. On peut également prévoir différentes versions de travail à partir de la même solution de départ pour représenter différents scénarios possibles. Nous permettons de sauvegarder chaque scénario (au format XML) indépendamment de la solution de départ (cf. listing 10.1). Pour cela, nous mémorisons pour chaque cellule libre (*GridCell*) représentée par son numéro de ligne et de colonne :

- l'indicateur de verrouillage (*CellLocked*)
- la forme urbaine affectée (*CellUrbanForm*)

L'utilisateur pouvant personnaliser le poids de chaque contrainte pendant l'interaction, ces éléments (*CompWeight*) doivent également être intégrés à la structure XML.

10.4 Paramétrage du comportement interactif

Le comportement des interactions entre le dispositif tactile et le solveur peut être modifié selon différents paramètres que l'utilisateur peut changer à tout moment. Certains paramètres agissent sur les prochaines saisies tandis que d'autres modifient le comportement du solveur. L'étude de ces différents paramètres permet de dresser un premier inventaire des fonctionnalités interactives mises à la disposition des opérateurs.

10.4.1 Comportement de saisie

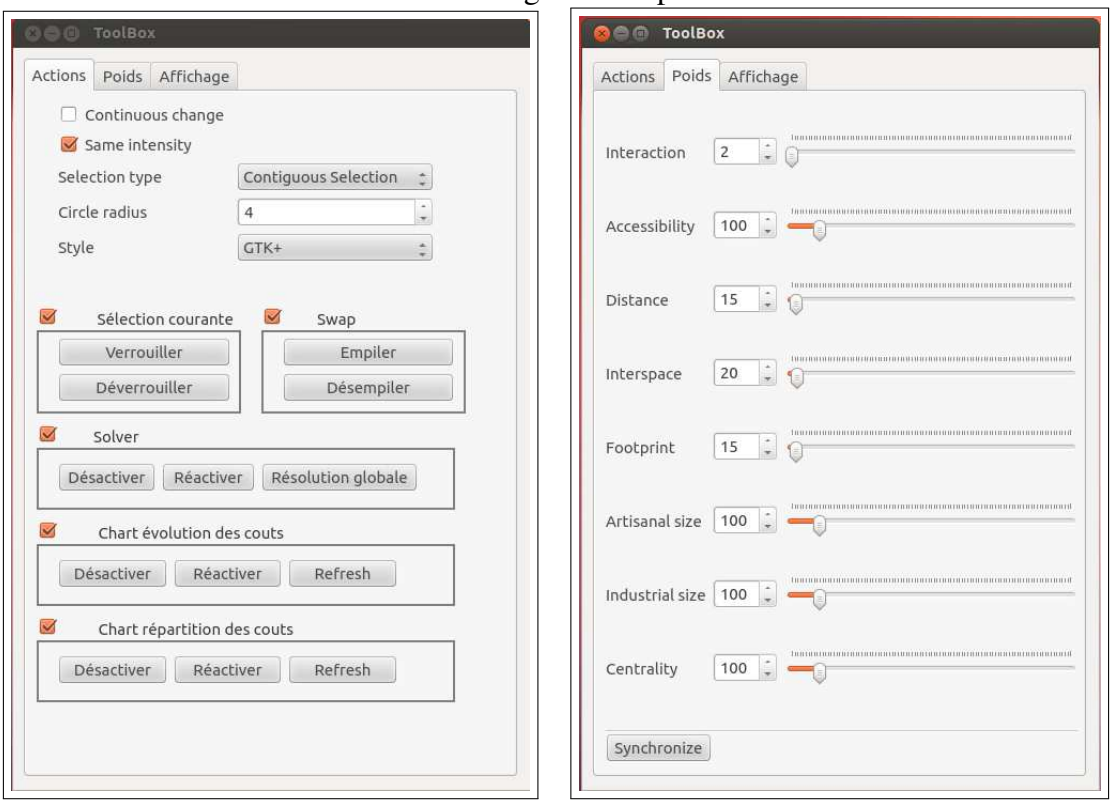
On propose aux utilisateurs quatre paramètres qui permettent de modifier le comportement de saisie sur l'IHM :

[3]Extensible Markup Language.

Listing 10.1 – Exemple de scenario mémorisé au format XML

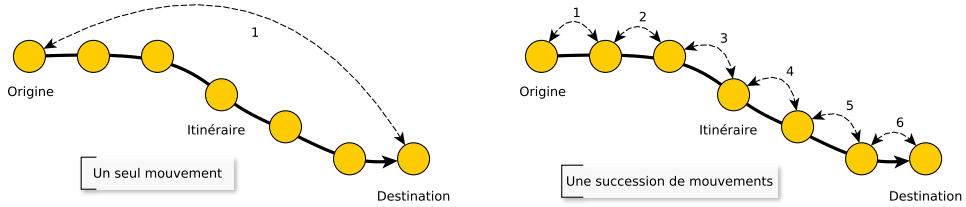
```
<?xml version="1.0" encoding="utf-8"?>
<Simulation>
  <File Name="/home/bruno/Documents/.../contrib-interaction/scenario-mlv-v14.xml"/>
  <GridCells>
    <GridCell CellRow="0" CellCol="0" CellLocked="false" CellUrbanForm="0"/>
    <GridCell CellRow="0" CellCol="1" CellLocked="false" CellUrbanForm="0"/>
    <GridCell CellRow="0" CellCol="2" CellLocked="false" CellUrbanForm="0"/>
    ...
    <GridCell CellRow="101" CellCol="256" CellLocked="false" CellUrbanForm="0"/>
    <GridCell CellRow="101" CellCol="257" CellLocked="false" CellUrbanForm="0"/>
    <GridCell CellRow="101" CellCol="258" CellLocked="false" CellUrbanForm="0"/>
  </GridCells>
  <CompWeights>
    <CompWeight CompIndice="0" WeightValue="2"/>
    <CompWeight CompIndice="1" WeightValue="100"/>
                ...
    <CompWeight CompIndice="8" WeightValue="100"/>
    <CompWeight CompIndice="9" WeightValue="1000"/>
  </CompWeights>
</Simulation>
```

FIGURE 10.3 – Paramétrage du comportement interactif.



- changement continu ;
- correspondance d'intensité obligatoire ;
- type de sélection ;
- taille du cercle d'influence.

FIGURE 10.4 – Mode continu désactivé (à gauche) ou activé (à droite).



10.4.1.1 Changement continu

Lorsque l'opérateur se saisit d'une forme urbaine affectée à une cellule[4] pour la déposer à un autre emplacement, soit on considère que l'opération se fait en un seul mouvement, soit on réalise une série de mouvements tout au long du déplacement en considérant l'itinéraire emprunté pour aller de l'emplacement d'origine jusqu'à l'emplacement de destination (voir la figure 10.4).

Un mouvement correspond à une permutation, la forme urbaine de la cellule d'origine étant échangée avec la forme urbaine de la cellule de destination. Si le mode continu est désactivé, une seule permutation est générée entre les cellules d'origine et de destination. Dans le cas contraire, le système tient compte de la trajectoire empruntée à partir de la cellule d'origine, et à chaque fois qu'on se déplace d'une cellule sur la trajectoire, une permutation est réalisée avec la cellule voisine. Après une série de permutations continues impliquant la forme urbaine prise à la source, celle-ci vient se loger sur la cellule de destination. Dans ce cas, toutes les formes urbaines rencontrées sur le parcours sont décalées d'une cellule vers l'emplacement d'origine. Si on considère par exemple la suite de formes urbaines suivante {A, B, C, D, E, F, G} pour aller de l'origine « A » à la destination « G », on obtient {**G**, B, C, D, E, F, **A**} hors mode continu alors qu'on obtient {**B, C, D, E, F, G, A**} si le mode continu est activé.

10.4.1.2 Correspondance d'intensité obligatoire

Si le paramètre « même intensité » est activé (ce qui correspond à la valeur par défaut), l'opérateur est obligé de sélectionner une cellule de destination dont l'intensité correspond à la cellule d'origine pour réaliser un mouvement. Néanmoins, si on rencontre des intensités différentes sur la trajectoire qui séparent les deux cellules origine et destination, en mode continu, ces dernières ne sont pas impliquées dans les échanges. Par exemple, si le trajet implique les formes urbaines {A, B, *C*, *D*, *E*, F, G} pour aller de l'origine « A » à la destination « G » et que les cellules liées à « C », « D » et « E » ont toutes des intensités différentes de l'origine « A », on obtiendra la suite {**B**, **F**, *C*, *D*, *E*, **G**, **A**}[5].

Cette option n'a aucun effet sur la contrainte de correspondance des niveaux d'intensité du solveur qui reste toujours active pour réduire les mouvements possibles initiés par la résolution.

10.4.1.3 Type de sélection

Le type de sélection peut correspondre à l'une des options suivantes :

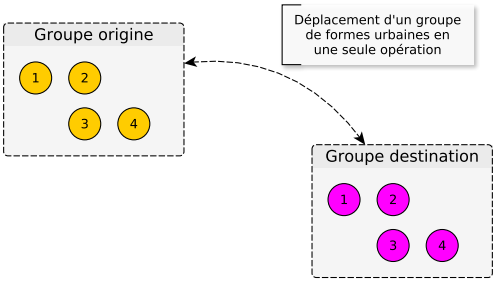
- sélection simple : une seule cellule peut être sélectionnée ;
- sélection étendue : pour sélectionner plusieurs cellules contiguës ou non ;
- sélection contiguë : pour sélectionner des cellules contiguës (en forme de rectangle).

Lorsque plusieurs cellules sont impliquées dans un même mouvement de groupe (le groupe correspondant

[4] En réalité, l'opérateur peut sélectionner plusieurs formes urbaines en même temps et le comportement reste le même pour une ou plusieurs formes urbaines.

[5] La même remarque peut être faite pour des cellules fixes présentes sur le parcours

FIGURE 10.5 – Déplacement d'un groupe de formes urbaines en une seule opération. Lorsque la correspondance d'intensité doit être respectée, il faut conserver une correspondance stricte entre les intensités des cellules d'origine et des cellules de destination, ces intensités étant repérées dans cet exemple par les chiffres 1, 2, 3, 4.



aux cellules sélectionnées au même moment par l'utilisateur), elles représentent toutes les cellules d'origine. Le processus de déplacement devient alors plus compliqué mais les principes évoqués jusqu'à présent restent applicables, que le mode continu soit activé ou non. Dans ce cas, un mouvement est autorisé uniquement avec un ensemble de cellules de destination ayant une correspondance stricte en nombre de cellules impliquées, en organisation spatiale (liée aux cellules sélectionnées) et en intensité (si la correspondance entre intensités est demandée) (voir figure 10.5).

10.4.1.4 Taille du cercle d'influence

Le cercle d'influence prend forme autour des cellules impliquées dans un déplacement et correspond à la fois à :

- un indicateur visuel permettant à l'opérateur de repérer les zones de la grille qui seront impactées par un mouvement ;
- une distance permettant au solveur de déterminer les zones de la grille à considérer autour des changements imposés par l'utilisateur pour exécuter une résolution locale limitée à ces zones.

La taille du cercle peut être paramétrée par l'opérateur pour agrandir ou réduire les zones impactées par un déplacement (voir figure 10.6). Plus le cercle est important, plus les effets d'un déplacement seront pris en compte sur la réorganisation automatique d'une ville en contre partie d'un temps de résolution plus long. L'utilisateur doit donc trouver un juste compromis entre la réactivité du système et la portée des réparations automatiques réalisées par le solveur, en fonction de son contexte de travail (puissance du matériel, temps d'attente admissible, etc.). D'autre part, il peut changer la taille du cercle à tout moment. Par exemple, l'utilisateur peut décider de fixer une taille du cercle en adéquation avec l'importance des opérations qu'il réalise, cette importance pouvant être liée au type des formes qu'il déplace, au nombre de formes urbaines impliquées dans un déplacement ou à la distance du déplacement.

10.4.2 Administration distante du solveur

Par l'intermédiaire de l'IHM, il est possible de :

- désactiver le solveur de base ;
- réactiver le solveur de base ;
- demander une résolution globale sur le solveur.

Par défaut, le solveur de base est actif. Lorsque l'opérateur demande sa désactivation, on conserve un échange des messages à la volée entre le dispositif tactile et le solveur interactif. Dans cette situation, le solveur interactif continue d'appliquer les permutations demandées par l'opérateur sur sa solution courante

FIGURE 10.6 – Cercle d'influence permettant d'identifier la taille des zones impactées par un mouvement. Dans cet exemple, le cercle de gauche identifie la zone d'origine et le cercle de droite la zone de destination, les cellules appartenant à ces deux zones étant soumises à une résolution locale. En dehors des cercles, les formes urbaines restent en place.

mais il ne demande plus aucune résolution locale, ce qui revient à une saisie manuelle coté IHM et à un report automatique des saisies sur la configuration courante du solveur interactif sans optimisation.

La réactivation du solveur de base intervient sur les prochains mouvements sans être rétroactive. Compte tenu de ce fonctionnement, il peut être utile de demander une résolution globale après avoir fait une série de modifications en mode désactivé. La résolution globale considère toutes les cellules de la grille comme sélectionnables. Par contre, les cellules qui ont été verrouillées par l'utilisateur restent verrouillées.

En associant ces différentes fonctionnalités, l'opérateur peut réaliser une série de modifications en maintenant une cohérence entre ses saisies et la configuration courante du solveur interactif, sans demander de résolution. Lorsque ses saisies sont terminées, il peut alors demander une résolution globale qui tiendra compte de tous les déplacements opérés et cherchera à réparer l'ensemble de la configuration courante sans remettre en cause les choix réalisés en bloc par l'opérateur.

10.4.3 Mode de fonctionnement autonome

Cette option intitulée « *empiler les permutations* » permet de travailler en mode autonome sur le dispositif tactile en interrompant jusqu'à nouvel ordre les communications réseau avec le solveur. Au lieu de transmettre, au fur et à mesure des saisies, les permutations imposées au solveur ; le système mémorise ces permutations dans une liste locale d'attente. Lorsque l'opérateur demande à « *désempiler les permutations* », la liste locale d'attente est transmise en un seul bloc au solveur qui applique en séquence toutes les permutations imposées. On revient alors à un mode de fonctionnement normal.

10.4.4 Administration des composants graphiques de retour

Il est possible de désactiver/réactiver la mise à jour temps réel des composants graphiques en charge de faire un retour visuel aux utilisateurs (cf. section 10.6). En effet, lorsque ces composants sont actifs, ils nécessitent des informations particulières en provenance du solveur après chaque résolution :

- coûts par contrainte ;
- coûts par cellule.

La désactivation permet d'alléger temporairement les échanges réseau entre le dispositif tactile et le solveur.

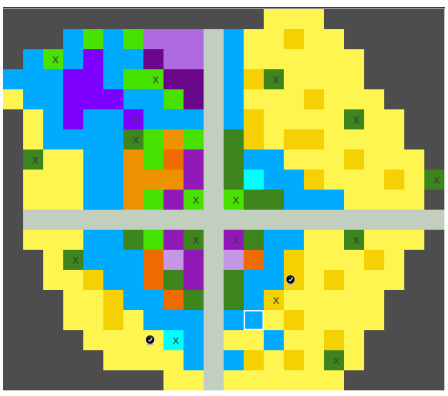
10.4.5 Réglage du poids de chaque contrainte

Côté modèle, un poids est spécifié pour chaque contrainte (voir la section 8.3.1). Grâce à ce dispositif graphique permettant d'ajuster dynamiquement les poids, l'utilisateur peut revenir sur les valeurs par défaut pour modifier la priorité donnée à chaque contrainte en fonction de ses propres appréciations (voir figure 10.3). Après la modification des poids sur le dispositif tactile, il faut demander une « synchronisation » qui se charge de transmettre ces nouveaux paramètres au solveur et qui demande en retour les nouveaux coûts par contrainte et par cellule pour réactualiser les composants graphiques de retour visuel lorsqu'ils sont actifs.

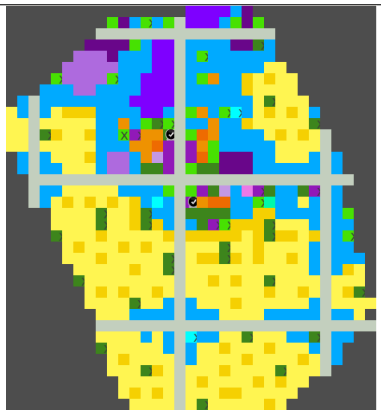
10.4.6 Taille de chaque cellule

Il s'agit de définir la taille graphique des cellules sur le dispositif tactile, chaque cellule correspondant toujours à un îlot urbain de 80x80 mètres. Cette fonctionnalité est très pratique pour représenter les villes importantes sur le dispositif tactile sans nécessiter trop de place à l'écran, ce qui permet à l'utilisateur de travailler à l'échelle graphique voulue, une petite taille étant propice à un déplacement groupé de plusieurs formes urbaines ou à des déplacements importants (en distance).

FIGURE 10.7 – Paramétrage de la taille des cellules.



Taille cellules = 30x30



Taille cellules = 14x14

10.5 Manipulations directes sur la grille

10.5.1 Déplacer une forme urbaine ou un ensemble de formes urbaines

Pour éviter qu'une modification locale réalisée par l'utilisateur puisse impacter la ville entière (i.e. changer l'ensemble de la solution), la manipulation d'un ensemble de formes urbaines a été définie de manière à impacter localement la carte. À cette fin, la taille du cercle d'influence est tout d'abord précisée par l'utilisateur (voir la section 10.4.1.4) pour définir la région dans laquelle les calculs (i.e. optimisation) seront effectués autour des changements demandés par l'utilisateur. Les utilisateurs peuvent alors sélectionner une ou plusieurs formes urbaines qu'ils déplacent interactivement sur la carte.

Deux modes d'interaction sont proposés :

- soit le calcul est effectué uniquement lorsque l'utilisateur dépose les formes urbaines sélectionnées à un nouvel emplacement ;
- soit le calcul est effectué en continu dès que l'utilisateur déplace les formes urbaines.

Dans le premier cas, les permutations imposées sont identifiées au moment où l'utilisateur dépose les formes urbaines. On demande alors au solveur une optimisation locale. Cette situation correspond au graphique de gauche de la figure 10.4 et ne nécessite qu'une seule demande d'optimisation.

Dans le second cas, à chaque mouvement (dans le sens d'un déplacement du pointeur sur la trajectoire), les permutations imposées sont définies et on demande une optimisation locale. Par rapport à la figure 10.4, ce cas est représenté par le graphique de droite et l'exemple nécessite six demandes successives d'optimisation, chaque demande correspondant à un mouvement sur l'itinéraire emprunté. Si on s'en tient aux traitements réalisés, le mode continu peut être ramené au premier cas : c'est comme si l'utilisateur avait demandé six *drag & drop* consécutifs en se déplaçant à chaque fois d'une seule cellule.

10.5.1.1 Construction de la LPI (Liste des Permutations Imposées)

Si le déplacement ne concerne qu'une forme urbaine, la construction de la LPI est simple car elle ne contient qu'une seule permutation liée aux deux cellules d'origine et de destination. Une fois la permutation récupérée coté solveur, les deux cellules sont identifiées et on procède à l'échange des valeurs entre les deux cellules.

Le problème est un peu plus complexe lorsque plusieurs formes urbaines sont impliquées dans un même déplacement. Dans ce cas, il faut projeter le déplacement sur les cellules de départ pour identifier une à une les cellules de destination. Par rapport à l'exemple à gauche de la figure 10.8, si l'utilisateur sélectionne les deux formes urbaines « 1 » et « 2 » du haut pour les déplacer vers le bas, on obtient les deux permutations suivantes :

- la permutation « A » correspond à la cellule d'origine (1,2) avec la cellule de destination (4,5) ;
- la permutation « B » correspond à la cellule d'origine (2,3) avec la cellule de destination (5,6).

Quel que soit l'ordre d'insertion des permutations dans la LPI, le solveur sera en mesure de placer les bonnes formes urbaines sur les bonnes cellules.

Néanmoins, si on considère l'exemple à droite sur la figure 10.8, la situation devient encore un peu plus difficile. Dans cet exemple, le déplacement concerne les deux cellules du haut positionnées en (1,2) et (2,3) mais l'utilisateur demande un décalage d'une seule cellule en descendant sur la diagonale. Nous avons toujours deux permutations à générer, mais dans le cas d'un chevauchement (intersection non vide) entre la zone d'origine et de destination, il faut tenir compte de la trajectoire pour déterminer l'ordre dans lequel les permutations imposées doivent être ajoutées à la LPI, pour qu'elles puissent être rejouées ensuite de façon cohérente sur le solveur. Si on considère les deux permutations suivantes :

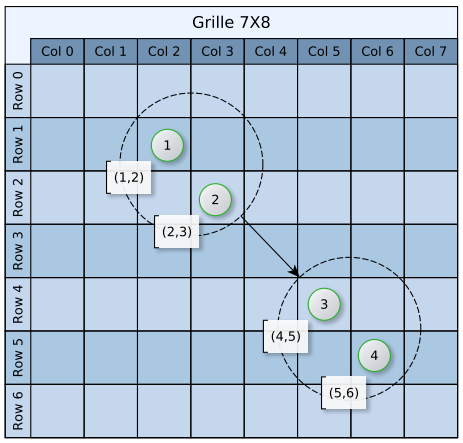
- la permutation « A » correspond à la cellule d'origine (1,2) avec la cellule de destination (2,3) ;
- la permutation « B » correspond à la cellule d'origine (2,3) avec la cellule de destination (3,4).

Il faut impérativement appliquer la permutation « B » avant la permutation « A » pour que la cohérence du déplacement soit maintenue. Lorsque le mode continu est sélectionné avec un déplacement groupé de plusieurs formes urbaines, ces situations sont habituelles puisque le système doit actualiser la LPI à chaque mouvement.

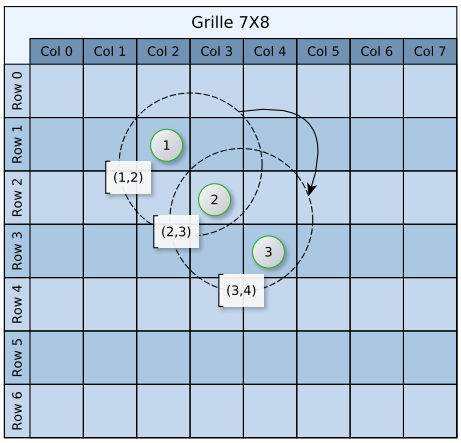
10.5.1.2 Prise en compte du cercle d'influence par le solveur

Au lieu de transférer vers le solveur interactif les identifiants des cellules présentes dans les cercles d'influence (représentés sur l'IHM), on transmet uniquement la dimension du cercle d'influence (le rayon) dans chaque message de type déplacement. C'est le solveur interactif qui se charge ensuite d'identifier les cellules appartenant aux cercles d'influence liés aux permutations imposées. Afin de simplifier les procédures, ces cellules correspondent à l'union des voisinages $\mathcal{V}^d_{l,c}$ de chaque cellule $V_{l,c}$ impliquée dans une permutation imposée, avec d correspondant au rayon du cercle (voir figure 10.9). Ces notations proviennent de la section 8.3.1.3.

FIGURE 10.8 – Création de la liste des permutations imposées en fonction du type de sélection et de déplacement des formes urbaines.

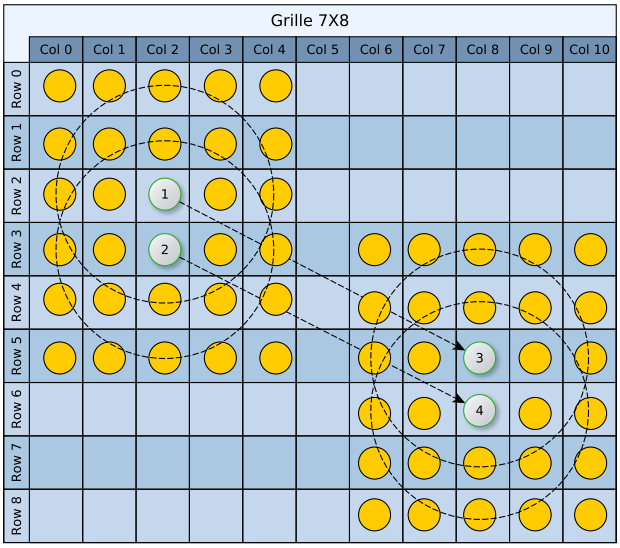


Ici, l'ordre d'application des permutations est sans importance



Lorsqu'il y a chevauchement entre la zone d'origine et de destination, il faut tenir compte de la trajectoire pour définir l'ordre d'application des permutations : échanger $2 \leftrightarrow 3$ puis $1 \leftrightarrow 3$

FIGURE 10.9 – Cercles d'influences interprétés sur le solveur interactif : lorsque l'utilisateur déplace (en groupe) les formes urbaines des cellules « 1 » et « 2 » sur les cellules « 3 » et « 4 » (en une seule opération), le solveur interactif identifie autour des cellules d'origine et de destination les cellules faisant partie des cercles d'influence. Pour cela, il utilise le voisinage $\mathcal{V}_{l,c}^{d}$ de chaque cellule $V_{l,c}$ impliquée dans une permutation imposée, avec d correspondant au rayon du cercle. Ces cellules sont alors marquées « sélectionnables » pour participer à l'optimisation locale.



10.5.1.3 Cohabitation entre actualisation automatique de la grille et manipulations utilisateur

Le système que nous proposons permet à l'opérateur de poursuivre ses manipulations sur l'interface graphique alors même que les résultats des optimisations demandées au solveur interactif :

- ne sont pas encore connus ;
- sont connus et éventuellement en cours de report sur l'IHM.

Pour autoriser ce fonctionnement particulier, plusieurs règles ont été élaborées :

- la solution courante présente sur le solveur interactif sert de référence : si il y a des différences entre la représentation de la solution courante sur l'IHM et celle du solveur interactif, c'est la solution présente sur le solveur interactif qui reste prioritaire ;
- l'IHM ne demande jamais au solveur de positionner une forme urbaine particulière sur une cellule. Au lieu de cela, elle demande au solveur d'échanger les formes urbaines présentent sur deux cellules données ;
- on s'autorise ponctuellement de légères différences entre les formes urbaines de la grille représentée sur l'IHM et la solution courante localisée sur le solveur ;
- le solveur transmet au dispositif tactile le résultat de ses optimisations en indiquant la forme urbaine associée à chaque cellule.

10.5.2 Verrouiller ou déverrouiller des cellules à la demande

À tout moment, l'utilisateur peut sélectionner un ensemble de cellules pour les verrouiller (sans avoir à réaliser de déplacement). Il peut également sélectionner un ensemble de cellules pour les déverrouiller.

10.6 Retours visuels

Les retours visuels sont prévus pour permettre aux utilisateurs d'évaluer les impacts de leurs décisions. Ces graphiques sont actualisés à la volée dès que des changements sont opérés sur la solution courante du solveur interactif.

10.6.1 Carte de chaleur

La carte de chaleur présente les régions sur la carte où les coûts sont les plus importants (voir la figure 10.10), la couleur rouge de la légende correspondant aux coûts les plus élevés.

Pour actualiser ce graphique, le solveur transmet simplement le coût de chaque cellule pour la solution courante. Les cellules étant situées sur la grille, on obtient instantanément une représentation spatialisée des coûts. Ces coûts sont directement disponibles dans notre modèle (cf. chapitre 8) grâce aux fonctions de coût définies pour chaque contrainte qui permettent de reporter les coûts des contraintes sur chaque tuple « variable | valeur ».

En mode continu, l'utilisateur peut suivre les impacts sur la carte en fur et à mesure qu'il déplace un groupe de formes urbaines, ce graphique étant actualisé à la volée à chaque mouvement.

10.6.2 Historique d'évolution des coûts

Une seconde représentation graphique, actualisée à intervalle de temps régulier (e.g. toutes les secondes), affiche le changement de coût global, ainsi que l'évolution des coûts pour chaque contrainte considérée (distance, accessibilité, taille critique, ...). Ainsi, l'utilisateur peut suivre l'impact de ses changements sur chaque contrainte et identifier les contraintes qui s'améliorent au fil du temps ou au contraire celles qui se dégradent (voir la figure 10.11).

FIGURE 10.10 – Carte de chaleur : les zones en rouge sur la carte font apparaître les régions correspondant aux coûts les plus élevés. Cette carte est directement actualisée avec les coûts de chaque variable disponibles dans notre modèle.

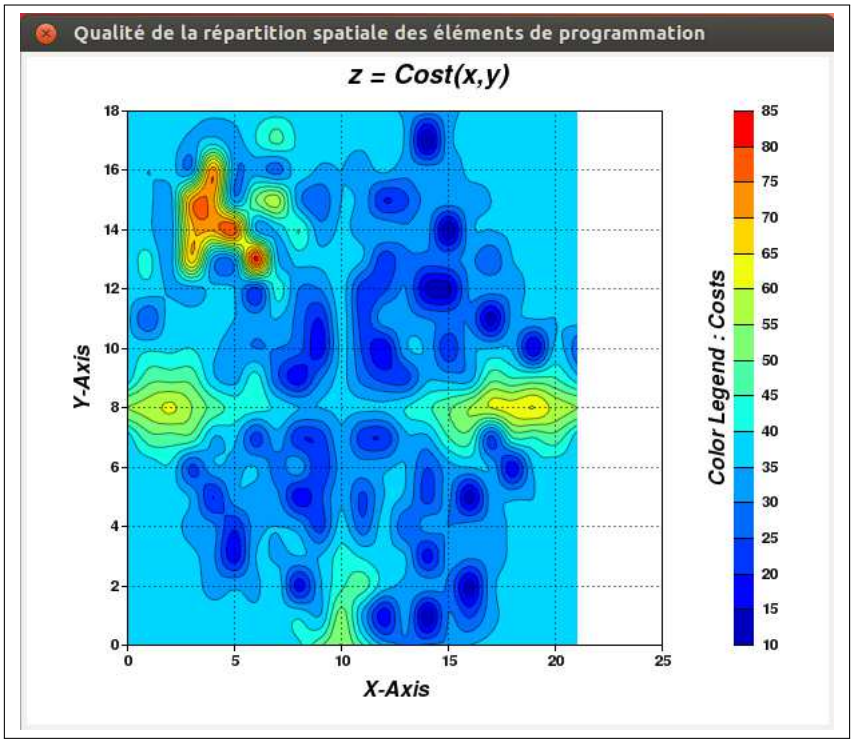
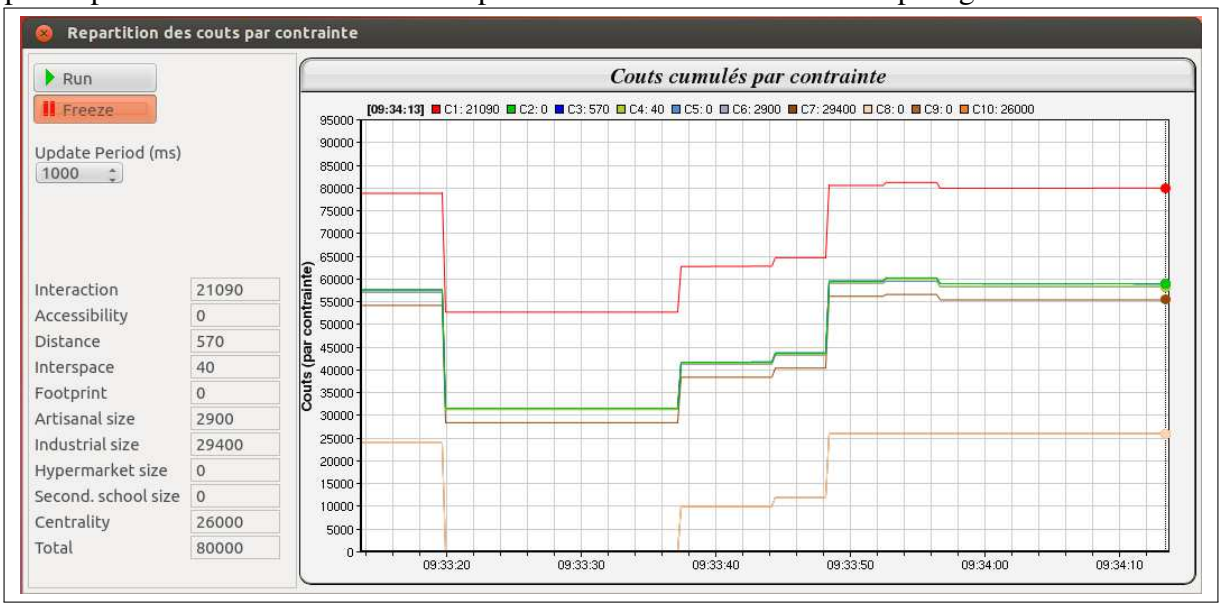


FIGURE 10.11 – Historique d'évolution des coûts par contrainte : ce graphique est composé de deux parties. La partie de gauche indique le coût lié à chaque contrainte pour la configuration courante tandis que la partie principale représente l'évolution des coûts par contrainte à intervalle de temps régulier.



A gauche du graphique, on retrouve un récapitulatif des coûts par contrainte lié à la dernière solution connue. La grille du milieu quant à elle permet de voir l'évolution des coûts de chaque contrainte en fonction du temps. Par défaut, ce graphique est rafraîchi à chaque seconde, cette intervalle de temps pouvant être modifié par l'utilisateur (actualisation toutes les 2 secondes, . . .). Dès qu'une modification est appliquée par le solveur sur la solution courante, il transmet les nouvelles informations de coût par contraintes qui seront

prises en compte au rafraîchissement suivant de l'écran.

10.7 Vidéo d'accompagnement

Certaines options du processus interactif de résolution sont détaillées dans la vidéo d'accompagnement (voir ici http://vimeo.com/80211470). En particulier, dans les situations où l'utilisateur déplace volontairement une partie d'une zone industrielle (pour laquelle une taille de zone critique est spécifiée), le processus de résolution repositionne automatiquement toutes les formes urbaines industrielles pour reformer une nouvelle zone industrielle pour la taille critique spécifiée en intercalant des espaces de respiration entre les usines et les maisons individuelles.

10.8 Conclusion

L'interaction entre composants est au cœur des systèmes complexes et, suivant la formule désormais bien connue, « *le tout qu'ils forment représente plus que l'addition des parties* ». Notre système d'aide à la décision n'échappe pas à cette formule et s'enrichit de l'interaction entre une IHM « taillée sur mesure » et un solveur interactif, qui communiquent entre eux par l'intermédiaire d'un gestionnaire de messages autonome.

Le système proposé maintient l'utilisateur au cœur du dispositif grâce à des fonctionnalités simples qui lui permettent de garder un contrôle total sur la solution finale. Il a également à sa disposition une série de paramétrages qui lui permettent d'adapter les comportements de l'interaction à ses propres besoins. Notre modèle (voir le chapitre 8) mis au point pour la résolution initiale est réutilisable dans ce contexte avec un minimum de modification, tout comme l'est le solveur de base utilisé pour la résolution initiale. En encapsulant le solveur de base dans un solveur interactif, ce dernier peut bénéficier de toute sa puissance tout en apportant de nouvelles fonctionnalités nécessaires à l'interactivité : le routage des messages, la détection d'événements, la mémorisation de la solution courante avec des traitements permettant de la manipuler, l'appel aux traitements d'optimisation, etc. La transformation des manipulations demandées par l'utilisateur en permutations permet de se conformer au fonctionnement habituel. On retrouve d'une part les permutations imposées par l'utilisateur sur l'IHM et d'autre part les permutations automatiques issues des optimisations réalisées par le solveur.

Le gestionnaire de messages asynchrones se révèle pertinent pour actualiser les informations de part et d'autre tout en conservant une IHM fluide et réactive en toutes circonstances. Pour que les résolutions puissent se faire à la volée sans trop s'écarter de la solution courante, un cercle d'influence est automatiquement défini autour des modifications opérées par l'utilisateur et seules les zones correspondantes sont concernées par le processus d'optimisation. Cette fonctionnalité est prise en charge par le solveur interactif qui ne rend sélectionnable, sur sa solution courante, que les cellules présentes dans les cercles d'influence. En réduisant la zone à (ré)optimiser, on réduit les temps de résolution et on améliore la solution courante sans trop s'en éloigner, la taille du cercle pouvant être modifiée pour ajuster l'éloignement autorisé et les temps de calcul.

11

Expérimentations

Dans ce chapitre, nous cherchons à évaluer différents aspects de nos travaux, à savoir les performances de nos différents algorithmes de résolution (voir le chapitre 9), la qualité des solutions générées à partir de notre modèle (voir le chapitre 8) ou encore la pertinence du mode interactif (voir le chapitre 10).

Pour faciliter la phase d'expérimentation, un paramétrage complet du solveur a été mis en place permettant de tester tel ou tel aspect de son fonctionnement en modifiant un fichier unique contenant les valeurs des différents paramètres. Parmi ces paramètres, on retrouve les entrées du modèle (avec les matrices d'interaction, etc.), la déclaration de chaque contrainte, le type de déploiement et des paramètres plus spécifiques comme la possibilité d'activer ou non la génération d'un tableau de bord complet. Le tableau de bord correspond à une série de sondes logicielles positionnées au sein du solveur qui, si elles sont activées, mémorisent au cours de la résolution des informations internes servant à générer des clichés instantanés sur son état (représentés par des graphiques et vidéos).

Afin d'évaluer l'efficacité de nos différents algorithmes, nous activons ou désactivons par des paramétrages adaptés certaines fonctionnalités proposées (e.g. version séquentielle ou distribuée, activation du mode multi-candidat, etc.). Nous comparons ensuite les temps nécessaires à une résolution initiale par rapport aux fonctionnalités mises en œuvre pour atteindre une qualité de solution qui soit acceptable pour un problème réel : *Marne-La-Vallée*[1].

Pour les deux autres aspects, à savoir la qualité des solutions et la pertinence de l'interaction, nous avons fait appel aux urbanistes impliqués dans le projet de recherche SUSTAINS pour avoir leurs retours. Pour évaluer la qualité, les expérimentations ont également été menées sur *Marne-La-Vallée*. Différentes cartes reprenant un nombre limité de formes urbaines ont été générées à partir de nos résolutions initiales, puis analysées pour vérifier l'emplacement des formes urbaines par rapport aux différentes zones d'intensité ou par rapport aux autres formes urbaines. Pour évaluer la pertinence de l'interaction, des démonstrations ont été menées avec trois urbanistes expérimentés pour recueillir leurs opinions. Une vidéo a également été mise à leur disposition pour représenter le fonctionnement général du mode interactif (cf. section 10.7).

11.1 Méthodologie

Dans la suite de ce chapitre, on parlera de *version* pour spécifier une exécution de la résolution initiale sur un seul processus (version mono-processus, voir section 9.3) ou distribuée sur plusieurs processus (version distribuée, voir section 9.4). On parlera de *mode* de fonctionnement pour indiquer si on supporte ou non

[1] Une ville nouvelle située à l'est de Paris, pour laquelle les urbanistes du projet SUSTAINS disposent de toutes les données utiles.

de multiples candidats (mode multi-candidat, voir section 9.3.3) et si on supporte ou non de multiples permutations (mode multi-permutation, voir section 9.3.4). On réservera le terme *protocole* pour savoir de quelle manière on exploite la version distribuée (protocoles 1 à 4, voir la section 9.4.3).

11.1.1 Versions incrémentales de nos prototypes

Le premier prototype de notre solveur (résolution initiale) a été développé en *Java* avec *CHOCO* [147] pour évaluer une méthode complète [222] de programmation par contraintes [16, 97, 9, 233]. Ce premier prototype ne donnant pas des résultats satisfaisants pour une version limitée de notre problème (voir la section 9.1 pour plus de détails), nous avons réalisé un second prototype en *Java* pour la même version du problème afin d'évaluer les métaheuristiques de voisinage [111] (cf. section 5.6.2), notamment les *méthodes de descente* (cf. section 5.6.4).

Compte tenu des résultats encourageants obtenus avec ce dernier prototype, la preuve du concept étant faite, nous nous sommes orientés vers une version opérationnelle, pour notre problème complet, réalisée en *C++* et basée sur le framework *EasyLocal++* [74] (voir le paragraphe 5.8.2). *EasyLocal++* a été retenu pour la puissance de son langage d'implémentation orienté objet, sa modularité et le fait qu'il soit dédié aux techniques de recherche locale, ce qui lui confère une grande cohérence et une facilité de prise en main. Dans la suite, nous considérons uniquement cette version opérationnelle.

11.1.2 Jeux de données

Nos jeux de données peuvent être exprimés sous la forme d'une *représentation simplifiée*, c'est à dire une représentation de la ville qui ne détermine qu'une partie des propriétés du problème (voir la section 11.1.2.1).

Afin de réaliser nos expérimentations, nous avons exploité différents jeux de données pour représenter les villes :

- neuf représentations simplifiées avec des tailles de grille différentes (voir la table 11.1) ;
- trois représentations complètes avec des tailles différentes : petite, moyenne, grande ;
- une représentation complète correspondant à la ville de Marne-La-Vallée (issue de données réelles liées à cette zone urbaine, fournies par les urbanistes du projet SUSTAINS).

TABLE 11.1 – Neuf jeux de données proposés avec une représentation simplifiée de la ville pour participer à nos expérimentations. Pour chaque taille de grille, nous mettons en correspondance le nombre de variables du problème (i.e. nombre de cellules libres).

Grille	Nombre de variables
8×8	64
16×16	256
24×24	576
32×32	1 024
40×40	1 600
48×48	2 304
56×56	3 136
64×64	4 096
128×128	16 384

11.1.2.1 Représentation simplifiée

Ce type de représentation ne détermine qu'une partie des propriétés du problème :

Listing 11.1 – Exemple de représentation simplifiée d'une ville constituée de 16×16 cellules, mémorisée au format XML.

```
<?xml version="1.0" encoding="utf-8"?>
<MainConfiguration>
  <Init>
  ...
    <Grid SizeX="16" SizeY="16"/>

    <IntensitiesByBlock Values="
      1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
      1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
      1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
      1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
      1 1 1 2 2 2 3 3 3 2 2 2 2 2 2 2
      1 1 1 2 2 3 3 3 3 3 3 2 2 2 2 2
      1 1 1 2 2 3 3 3 3 3 3 3 2 2 2 2
      1 1 1 2 3 3 3 3 3 3 3 3 3 2 2 2
      1 1 1 2 3 3 3 3 3 3 3 3 3 2 2 2
      1 1 1 2 3 3 3 3 3 3 3 3 3 2 2 2
      1 1 1 2 2 3 3 3 3 3 3 3 2 2 2 2
      1 1 1 1 2 2 3 3 3 3 3 2 2 2 2 2
      1 1 1 1 1 1 2 2 3 3 3 2 2 2 2 2
      1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1"/>

    <UrbanForms>
      <UrbanForm Id="1" Description="Maison individuelle">
        <Allocation Intensity="1" NumberOfBlocks="38"/>
      </UrbanForm>
      ...
      <UrbanForm Id="19" Description="Espace de respiration">
        <Allocation Intensity="1" NumberOfBlocks="5"/>
        <Allocation Intensity="2" NumberOfBlocks="8"/>
        <Allocation Intensity="3" NumberOfBlocks="5"/>
      </UrbanForm>
    </UrbanForms>
  ...
  </Init>
</MainConfiguration>
```

- la taille de la grille (16×16) (balise « *Grid* ») ;
- l'intensité de chaque cellule (balise « *IntensitiesByBlock* ») ;
- le nombre de chaque catégorie de forme urbaine à ventiler sur la ville, par niveau d'intensité (balise « *UrbanForms* »).

Avec ces représentations, il n'y a pas de centralité et toutes les cellules font partie d'une ville carrée ou rectangulaire. Les représentations simplifiées peuvent être construites manuellement. Par exemple, le listing 11.1 fournit une représentation simplifiée pour une grille 16×16.

11.1.2.2 Représentation complète

Les représentations complètes incorporent toutes les données du problème, y compris des informations nécessaires aux applications des différents partenaires impliqués dans le projet SUSTAINS. On y retrouve des éléments généraux de paramétrage (formes urbaines, axes routiers principaux, ...) et une information détaillée pour chaque cellule :

- coordonnées spatiales ;
- indicateur (o/n) pour savoir si la cellule est présente ou non dans la ville ;
- indicateur (o/n) pour savoir si la cellule correspond ou non à une centralité ;
- indicateur (o/n) pour savoir si la cellule est constructible ;
- forme urbaine fixe liée à la cellule (route, rivière, ...) ;
- élévation de la cellule ;
- etc.

Les représentations complètes sont générées automatiquement par les logiciels du LIMSI[2], un partenaire du projet SUSTAINS en charge d'identifier les propriétés d'une ville (contours, zones d'intensité, avenues principales, etc.). La masse des informations présentes dans cette représentation est telle qu'une construction manuelle de ce fichier n'est pas envisageable. Notre solveur[3] récupère en entrée les éléments qui lui sont utiles et incorpore le résultat de ses optimisations dans une copie du fichier avant de la transmettre aux partenaires, notre objectif étant de renseigner une forme urbaine pour chaque cellule libre.

11.1.3 Paramètres de configuration

Le comportement du solveur dépend de nombreux paramètres que nous avons choisi de structurer et de regrouper dans un fichier de configuration[4] (au format XML). Plusieurs fichiers avec des paramètres particuliers ont été créés pour définir des configurations différentes, ce qui permet de choisir un fichier de configuration spécifique au lancement du solveur pour qu'il utilise des paramètres particuliers.

11.1.3.1 Indépendance des paramètres par rapport au problème à résoudre

Pour que les paramètres de configuration soient indépendants du problème à résoudre, deux approches complémentaires ont été mises en place :

- certains paramètres sont exprimés en pourcentage pour les rendre dynamiques (e.g. taille initiale de la liste des candidats bannis) ;
- la description du problème à résoudre (sous une forme simplifiée ou complète) dépend d'un fichier XML indépendant du fichier de configuration.

L'utilisateur doit donc préciser deux éléments au lancement d'une résolution :

- l'emplacement et le nom du fichier qui contient les paramètres de configuration ;
- l'emplacement et le nom du fichier qui contient la représentation du problème.

11.1.3.2 Tirage de nombres aléatoires

Un paramètre de configuration particulier concerne le tirage de nombres aléatoires. Il s'agit du paramètre « *SeedValue* » qui prend une valeur entière positive ou nulle. Si $SeedValue = 0$, une même séquence de tirages aléatoires donnera des résultats différents. Dans le cas contraire, pour une valeur donnée de $SeedValue$, une même séquence de tirages aléatoires donnera les mêmes résultats (i.e. une série de nombres aléatoires identiques). Par exemple, ce paramètre permet de générer (ou non) des solutions aléatoires suivant une même séquence de tirages aléatoires : ceci est essentiel lorsque l'on veut comparer des résultats en partant d'une solution de départ identique.

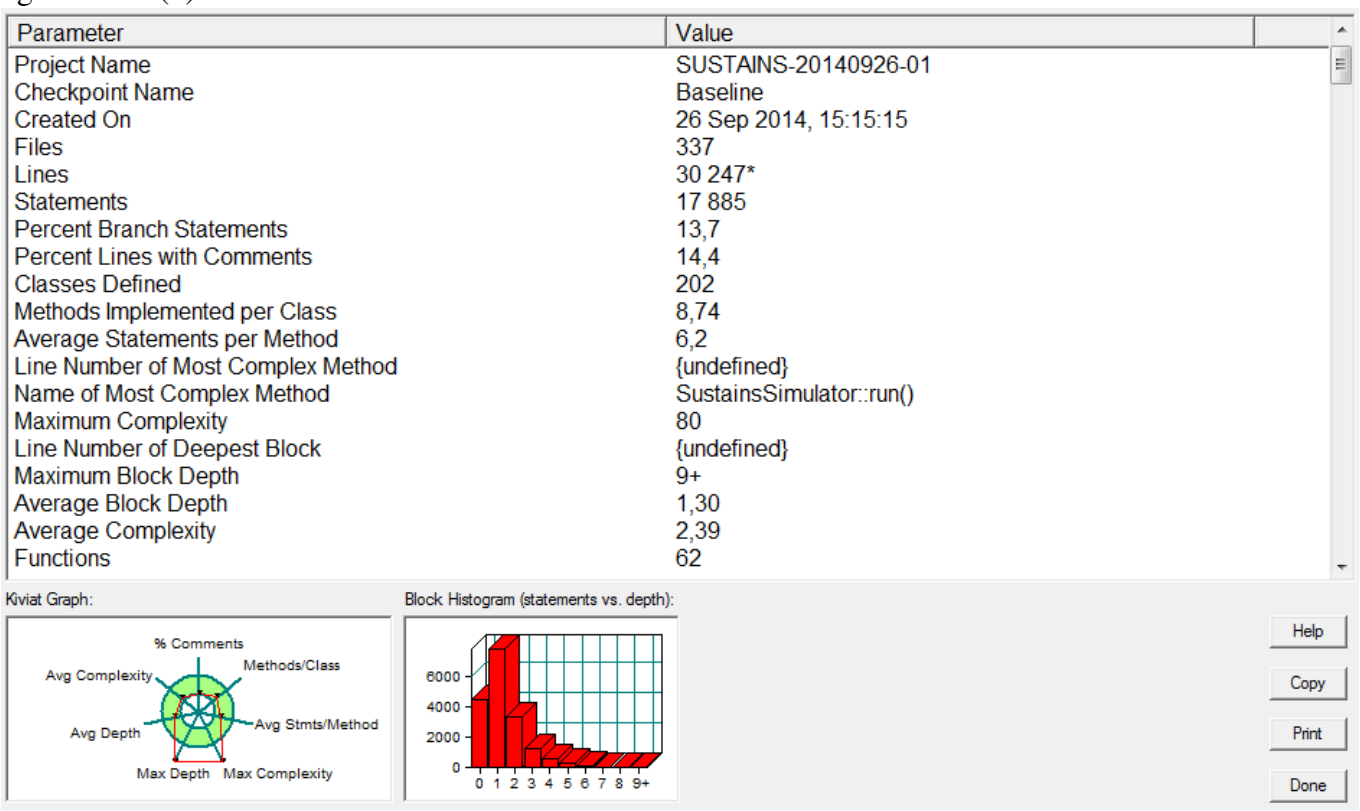
En effet, les données indiquées dans nos expérimentations correspondent à une moyenne de 10 exécutions, chaque exécution étant réalisée avec une valeur différente pour $SeedValue$ que l'on notera $\{seed1, seed2, \ldots, seed10\}$. De cette façon, chaque exécution démarre avec une solution initiale aléatoire

[2] Le Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI) http://www.limsi.fr/index.fr.html.

[3] Lorsqu'aucune précision n'est donnée, cela concerne indifféremment le solveur de résolution initiale et le solveur interactif.

[4] Il s'agit ici d'un fichier de paramétrage, à ne pas confondre avec une configuration dans le sens solution à un problème.

FIGURE 11.1 – Résumé des mesures sur l'application développée. Généré par l'outil *SourceMonitor* (http://www.campwoodsw.com/sourcemonitor.html). Le nombre de lignes indiqué écarte les lignes vides (*).



différente. Lorsque l'on veut comparer plusieurs versions (mono-processus ou distribué) ou modes de fonctionnement (multi-candidats, multi-permutation) différents, chaque série d'exécutions reprend la même séquence de valeurs pour $SeedValue$, correspondant à la même séquence de solutions de départ.

11.1.4 Implémentation et déploiement

11.1.4.1 Implémentation

L'application a été développée en *C++* et s'appuie sur le framework *EasyLocal++* [74]. La communication entre les processus distants se fait par échanges de messages en utilisant le middleware orienté message *Apache ActiveMQ* (http://activemq.apache.org/) en lien avec la librairie *ActiveMQ-CPP* (http://activemq.apache.org/cms/using-activemq-cpp.html).

Cette communication distante intervient à deux niveaux :

- en version distribuée : pour gérer les communications entre le processus maître et ses esclaves ;
- en mode interactif : pour gérer les communications entre le dispositif tactile et le solveur interactif.

La figure 11.1 fournit un résumé de l'application sur la base de métriques standards. Il s'agit d'un développement conséquent qui définit un peu plus de 200 classes *C++* pour 30 000 lignes de code. Cette application fonctionne sur les systèmes d'exploitation *Windows* et *Linux*. Nous utilisons le service *InriaForge* pour faciliter le travail collaboratif, notamment *Apache Subversion* (https://subversion.apache.org/) comme gestionnaire centralisé de versions pour le code source de l'application. Il s'agit en effet d'un travail d'équipe réparti entre les sites de Nantes et de Rennes, avec une partie des développements confiée à un ingénieur de l'Université Rennes 1 qui a pris en charge :

- le support complet des données XML ;

- les développements PHP[5] pour gérer les échanges via HTTP avec les applications des partenaires, notre moteur de résolution initiale étant exposé comme un service accessible via le Web ;
- la construction des principales IHM de rendu (en dehors des parties liées au mode interactif, cf. chapitre 10).

11.1.4.2 Déploiement

Nous avons opéré trois types de déploiement différents et complémentaires sur des machines Linux.

Station de travail Dans un premier temps, les différentes versions sont déployées pour être testées sur ma station de travail portable équipée de 3,8 Gio de mémoire et d'un processeur multi-cœurs Intel® Core™ i7-2630QM CPU @ 2.00GHz x 8. Le système d'exploitation correspond à la version *Ubuntu* 12.04 (precise) 64 bits.

Serveur de l'INRIA à Rennes Une version stable du solveur est déployée sur un serveur de l'INRIA à Rennes pour assurer un service permanent de *résolution initiale* ouvert aux partenaires du projet SUSTAINS. La communication se fait par échange, via HTTP[6], du fichier qui décrit la ville d'étude (voir la section 11.1.2.2 qui décrit la représentation complète).

Grid'5000 Concernant la version distribuée, elle est déployée sur la plate-forme française *Grid'5000*[7], une grille de calcul expérimentale dédiée à la recherche pour réaliser des calculs distribués à grande échelle.

11.2 Tableau de bord

Nous avons instrumentalisé le code source de nos applications pour pouvoir analyser après coup le fonctionnement interne de notre solveur et le comportement des différentes contraintes. Les analyses fournies par le tableau de bord correspondent à des graphiques et à une vidéo, générés à la suite d'une résolution initiale. Ces analyses ont permis d'adapter notre modèle lorsque cela était nécessaire (e.g. problème de performance avec la contrainte globale de regroupement sur des instances de taille importante : grille 64x64). Elles peuvent également être utilisées pour améliorer les performances du solveur ou procéder aux réglages des nombreux paramètres de lancement utilisés lors des expérimentations.

Les graphiques repris dans cette section ont pour la grande majorité été réalisés à partir d'une résolution initiale lancée le 7 février 2013 à 16h10 en utilisant une représentation incomplète basée sur notre grille 32×32. Il s'agit de données de tests qui ne correspondent pas nécessairement à la réalité, et qui peuvent être en décalage avec la dernière version de notre modèle (voir la section 8.3.1). Cette résolution était liée au paramétrage suivant des contraintes :

- Interaction entre formes urbaines : poids = 1 ;
- Accessibilité en tout point : poids = 100 ;
- Éloignement : poids = 15 ;
- Espace ou équipement tampon : poids = 20 ;
- Emprise (place) : poids = 15 ;
- Taille mini zone activité artisanale : taille minimum d'un groupe = 6 et poids = 1 ;
- Taille mini zone activité industrielle : taille minimum d'un groupe = 12 et poids = 1.

[5] Hypertext Preprocessor : langage de programmation principalement utilisé pour produire des pages Web dynamiques (http://php.net/).

[6] Hypertext Transfer Protocol en anglais ou protocole de transfert hypertexte.

[7] **Remerciements** : plusieurs expériences présentées dans cette thèse ont été réalisées en utilisant la plateforme expérimentale Grid'5000, issue de l'Action de Développement Technologique (ADT) Aladdin pour l'INRIA, avec le support du CNRS, de RENATER, de plusieurs Universités et autres contributeurs (voir https://www.grid5000.fr)

Listing 11.2 – Paramètres de configuration : exemple de paramétrage des sondes logicielles intégrées au solveur. La totalité des sondes peut être désactivée (*OverallActivation*). L'activation/désactivation peut également intervenir au niveau de chaque sonde (*isActivated*).

```
<Observers OverallActivation="false">
  <Observer Id="BestMoveObserver" Description="Meilleur score trouve (a chaque iteration)"
          IsActivated="true"/>
  <Observer Id="DeltaCostObserver" Description="Temps moyen par cycle (iteration)"
          IsActivated="true" NbEventByCycle="20000" NumEventStart="0" NumEventStop="0"/>
  <Observer Id="MakeMoveGUIObserver" Description="Permutation appliquee" IsActivated="true"/>
  <Observer Id="RunnerObserverStartEnd" Description="Duree/cout au debut/fin de resolution"
          IsActivated="true"/>
  <Observer Id="RunnerObserverNewBest" Description="Duree/deltaCost global et par contrainte"
          IsActivated="true"/>
  <Observer Id="ScoreLandscapeObserver" Description="Paysage des deltaCosts par cellule"
          IsActivated="true"/>
</Observers>
```

D'autre part, il y avait 1 024 formes urbaines à répartir sur le territoire, ce nombre étant ventilé de la façon suivante (ventilation fournie ici sans reprendre le détail par niveau d'intensité) :

- 152 maisons individuelles ;
- 120 maisons de ville ;
- 88 habitats intermédiaires ;
- 48 habitats collectifs R+4 ;
- 40 habitats collectifs R+7 ;
- 80 bâtiments tertiaires R+4 ;
- 40 bâtiments tertiaires R+7 ;
- 140 activités artisanales ;
- 80 activités industrielles ;
- 32 centres commerciaux ;
- 16 écoles maternelles ;
- 20 écoles primaires ;
- 20 collèges ;
- 12 lycées ;
- 12 équipements administratifs ;
- 12 équipements techniques ;
- 20 espaces sportifs de quartier ;
- 20 espaces sportifs inter quartier ;
- 72 espaces de respiration.

11.2.1 Mode opératoire

La génération des éléments du tableau de bord est prise en charge par des sondes logicielles positionnées au cœur du solveur (voir un exemple de paramétrage avec le listing 11.2). Lorsqu'elles sont activées, elles génèrent automatiquement des fichiers de trace pendant la phase de résolution. A la fin du processus de résolution, des scripts sont disponibles pour générer les différents graphiques ainsi qu'une vidéo d'évolution des paysages (voir la section 11.2.5). Pour retrouver les valeurs des principaux paramètres de configuration utilisés lors de l'expérimentation, un fichier est généré afin d'en garder une trace. Tous les résultats sont stockés dans un répertoire dont le nom est dynamique, en tenant compte de la date et de l'heure de lancement.

11.2.2 Variation des coûts

Chaque permutation (i.e. échange des valeurs entre deux cellules de la grille, voir la section 8.3.1) appliquée sur une configuration impacte son coût global en plus ou en moins. Cette différence correspond à la *variation de coût* relative à chaque permutation appliquée sur une configuration courante (i.e. à chaque itération de notre algorithme de résolution). Nous verrons dans les sections suivantes qu'une variation de coût peut également être calculée pour une permutation sans qu'elle soit appliquée sur la configuration courante. Dans ce cas, on parlera d'une *variation de coût potentielle*.

Les graphiques de variation des coûts que nous proposons se présentent sous deux formes :

- une partie récapitulative mettant en parallèle les différentes contraintes sur le même graphique (voir la figure 11.2).
- une partie détaillée avec un graphique différent par contrainte (voir la figure 11.3) ;

A chaque itération, le système indique la variation du coût enregistrée pour chaque contrainte. La plupart du temps, la variation est négative, ce qui contribue à la réduction du coût global sur la configuration courante (nota : les variations apparaissent inversées sur nos graphiques). Il arrive néanmoins que la variation soit positive sur une ou plusieurs contraintes. Ceci traduit une situation de conflit entre les différentes contraintes où le gain obtenu pour l'une se fait au détriment des autres, le solde permettant tout de même de réduire le coût global.

A la lecture des exemples proposés (voir la figure 11.2 ou 11.3), on voit une très forte activité tout au long du processus de résolution pour les contraintes d'interaction (voir la section 8.3.2.3), d'éloignement (voir section 8.3.3.1) et d'espace ou équipement tampon (voir section 8.3.3.4). Ceci paraît normal pour la contrainte d'interaction qui est une contrainte générale qui s'applique à l'ensemble des cellules (voir la section 8.3.2). Les deux autres contraintes (éloignement et espace ou équipement tampon) sont des contraintes spécifiques (voir la section 8.3.3), néanmoins elles s'appliquent à beaucoup de formes urbaines et tout comme la contrainte d'interaction, elles concernent le voisinage d'un nombre important de cellules. La contrainte d'éloignement est néanmoins plus active au début de la résolution que la contrainte d'espace ou équipement tampon qui présente une plus forte activité au milieu de la résolution. On peut en déduire qu'une fois les formes urbaines suffisamment éloignées les unes des autres, elles rentrent moins en conflit avec la contrainte d'éloignement. En fait, ces trois contraintes sont étroitement liées les unes aux autres, la modification de l'une ayant des incidences directes sur les deux autres.

La contrainte d'accessibilité (dans sa version locale, voir la section 8.3.3.3) travaille beaucoup au tout début de la résolution, ceci étant provoqué par son poids, le plus important parmi l'ensemble des contraintes, qui la rend prioritaire sur les autres. Et une fois que les espaces de respiration sont bien répartis sur le territoire, cette contrainte n'est que très rarement impactée.

Les deux contraintes de taille minimum qui s'appliquent aux zones d'activité artisanale et industrielle présentent une activité différente l'une par rapport à l'autre. Il s'agit pourtant de la même contrainte appliquée à deux formes urbaines distinctes ! Ceci peut s'expliquer, d'une part, par des paramètres de regroupement différents, le nombre de formes industrielles à regrouper (nombre minimum pour former un groupe) étant plus important que celui des activités artisanales. Une fois le regroupement des industries réalisé, il est donc plus difficile à remettre en question. D'autre part, les relations entre les activités artisanales et les autres formes urbaines sont plus souples (moins contraignantes) que celles liées aux industries, ce qui fait que les industries sont plus vite isolées du reste, et moins impactées au fil de la résolution.

La contrainte d'emprise (formation des places, voir la section 8.3.3.5) est la contrainte qui est de loin la moins active. Malgré son activité très faible, les variations de coût qu'elle présente sont globalement importantes à chaque fois qu'elle est impactée. Cette contrainte intervient uniquement autour des collèges et lycées, elle est donc limitée en nombre de cellules. Ceci explique probablement pourquoi elle est rarement impactée.

FIGURE 11.2 – Récapitulatif des variations de coût par itération pour chaque contrainte. Les variations de coût qui apparaissent sur l'axe des ordonnées sont inversées sur les graphiques, de sorte qu'une variation négative apparaît au dessus de l'axe des abscisses, l'axe des abscisses représentant l'évolution du nombre des itérations au cours d'une résolution.

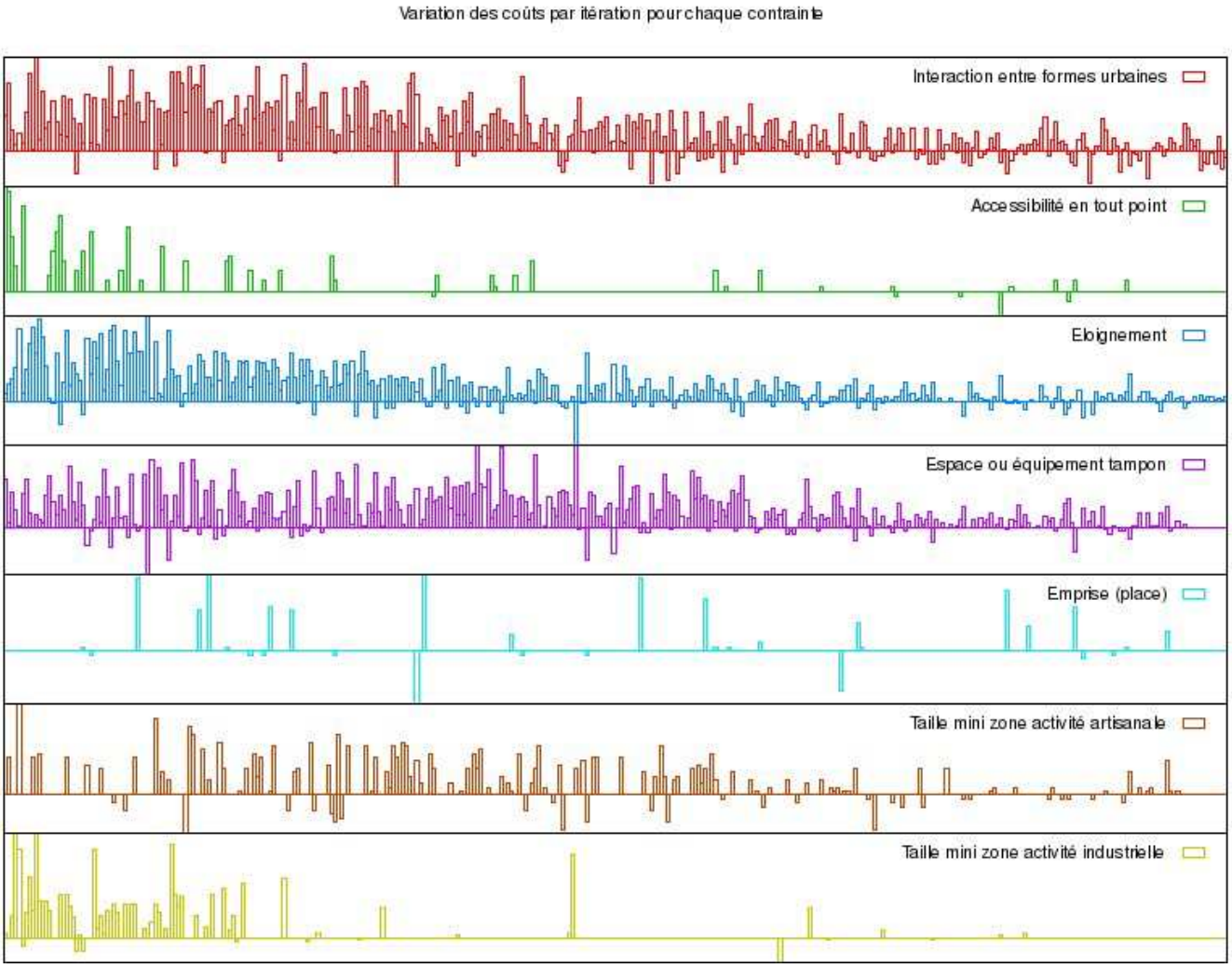
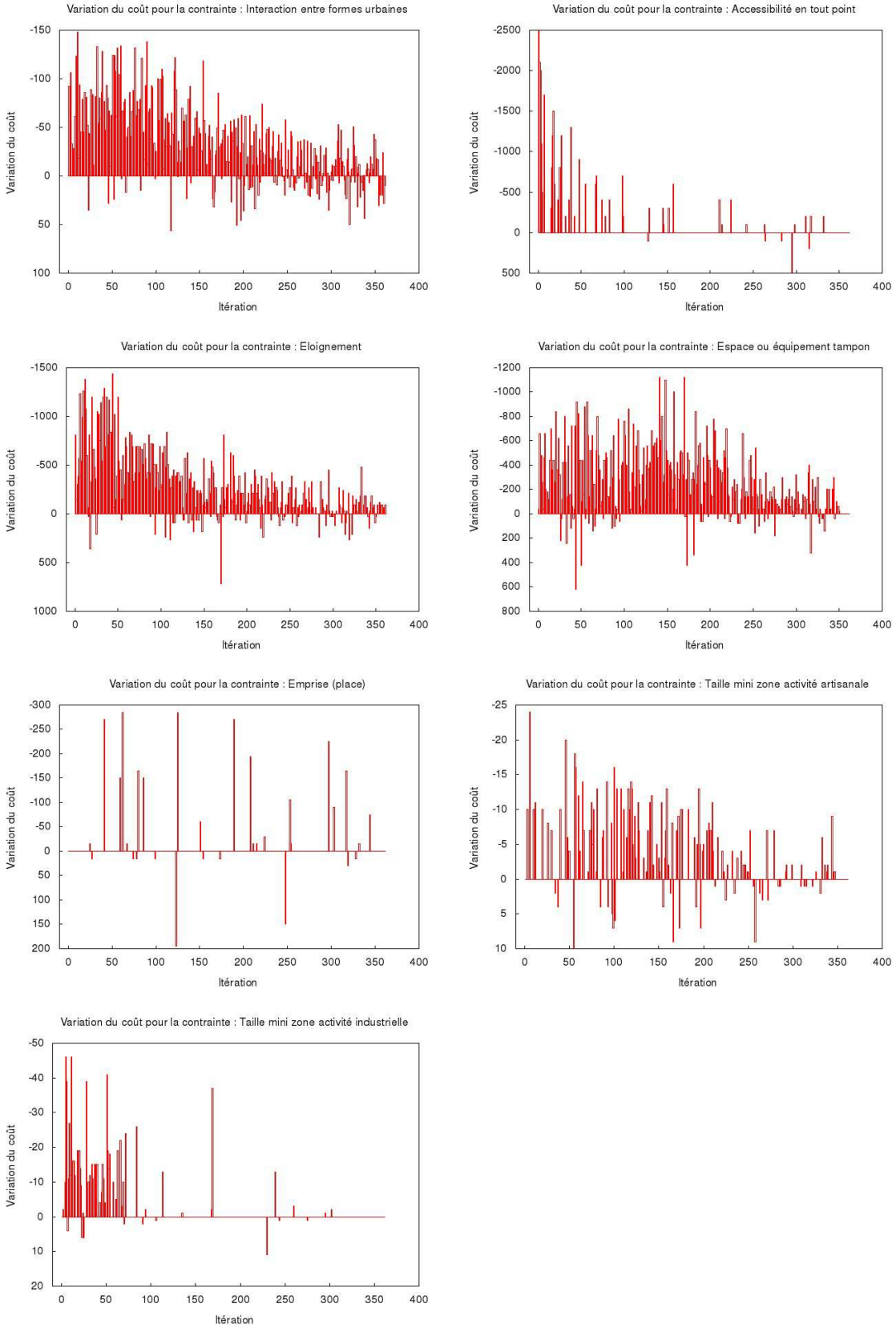


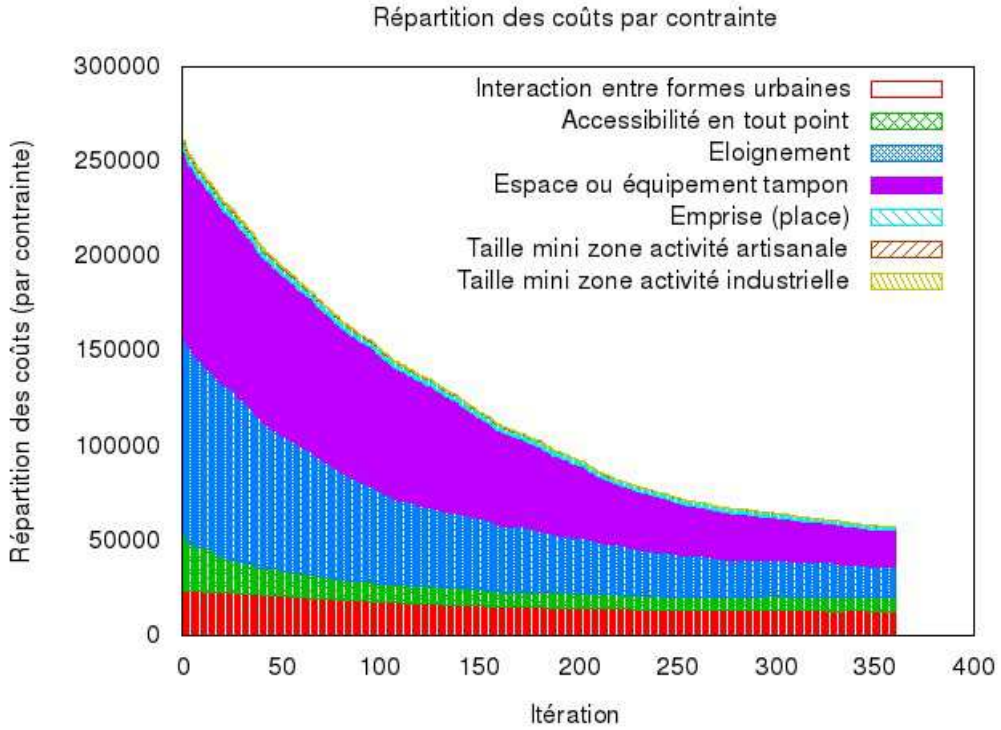
FIGURE 11.3 – Détail des variations de coût par itération pour chaque contrainte. Les variations de coût qui apparaissent sur l'axe des ordonnées sont inversées sur les graphiques, de sorte qu'une variation négative apparaît au dessus de l'axe des abscisses, l'axe des abscisses représentant l'évolution du nombre des itérations au cours d'une résolution.



11.2.3 Répartition des coûts par contrainte

Le graphique de répartition des coûts par contrainte permet de visualiser au fil des itérations la réduction des coûts de chaque contrainte et la proportion qu'occupe chaque contrainte dans le coût global de la configuration courante (répartie dans le temps). Par rapport à l'exemple de la figure 11.4, les coûts les plus importants au début de la résolution sont répartis entre les deux contraintes *Éloignement* et *Espace ou équipement tampon*. Ce sont également ces deux contraintes qui voient leurs coûts respectifs diminuer le plus par rapport aux autres contraintes, ces diminutions étant les plus importantes au milieu de la résolution. Concernant la contrainte *Accessibilité en tout point*, elle subit une diminution importante au tout début du processus de résolution. La même remarque peut être faite pour la contrainte *Interaction entre formes urbaines* qui diminue toutefois moins fortement et met plus de temps à se stabiliser, son coût restant globalement constant à partir de l'itération 150. Les coûts des autres contraintes (Emprise, Taille minimum zone d'activité artisanale et industrielle) restent très faibles tout au long de la résolution. Ceci peut s'expliquer par un nombre limité de formes urbaines concernées par ces contraintes. Pour leur donner plus d'expressivité, on pourrait fixer des poids supérieurs aux contraintes de regroupement (taille minimum), ce qui permettrait de rééquilibrer la proportion des coûts entre l'ensemble des contraintes (voir la section 8.3.1).

FIGURE 11.4 – Répartition des coûts par contrainte : à chaque itération (en abscisse), on superpose le coût total de chaque contrainte (en ordonnée). On reconstitue ainsi le coût global de la configuration courante qui diminue au fil de la résolution. La représentation correspond à un histogramme très resserré, on ne peut donc pas distinguer les barres qui sont réduites à des lignes verticales dont les couleurs correspondent aux rectangles de la légende : la première forme urbaine indiquée dans la légende correspond à la forme urbaine positionnée en bas du diagramme, etc.

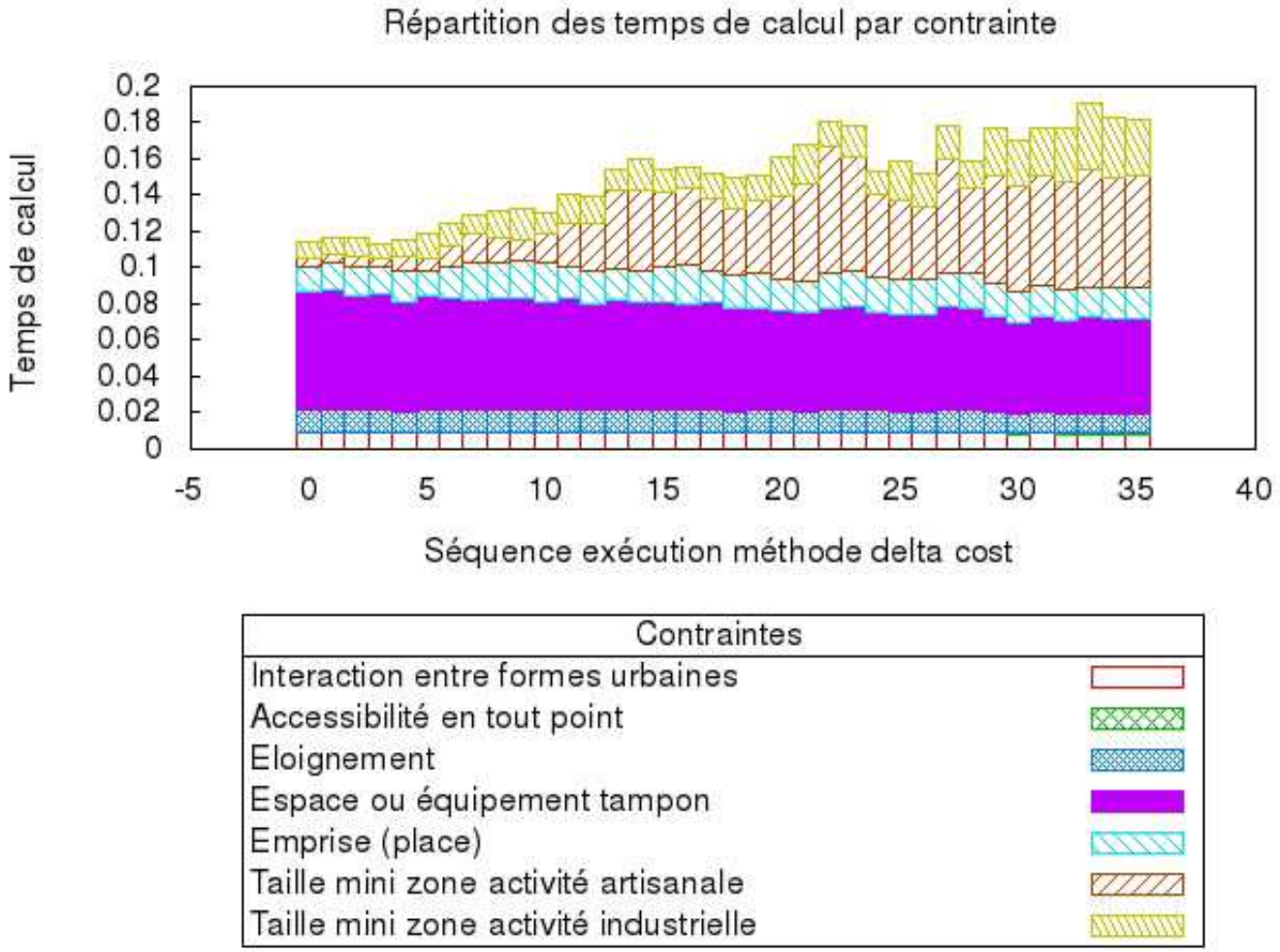


11.2.4 Répartition des temps de calcul par contrainte

On s'intéresse ici à la performance des fonctions qui calculent, pour chaque contrainte, la variation du coût occasionnée par les permutations potentielles ou par les permutations effectivement appliquées. Le nombre d'évaluations étant très important (pour les *variations de coût potentielles*, voir le paragraphe 7.5.2.2), on

calcule une moyenne des temps de calcul pour un nombre d'appels donné. Par exemple, tous les 20 000 appels à ces fonctions, on enregistre le temps moyen requis pour chaque contrainte et on recommence pour les 20 000 appels suivants, etc. Les temps sont exprimés en millisecondes avec une représentation cumulée pour chaque contrainte. Sur la figure 11.5, on démarre la résolution avec un temps moyen cumulé de 0,11 ms et on la termine avec un temps moyen cumulé à 0,18 ms. L'analyse de ce graphique permet d'expliquer cette augmentation du temps de calcul moyen. En effet, au fur et à mesure de la résolution, le temps moyen de calcul pour la contrainte *Espace ou équipement tampon* diminue très légèrement alors que le temps moyen de calcul lié aux contraintes de *taille minimum* augmente plus fortement, avec une augmentation supérieure pour la contrainte de taille minimum liée aux zones d'activité artisanale. Le nombre de formes urbaines à répartir pour les activités artisanales est supérieur à celui des activités industrielles (140 contre 80) alors que la taille minimale pour former une zone d'activité artisanale est inférieure à celle des activités industrielles (6 contre 12). Ceci est suffisant pour expliquer le surcroît de temps nécessaire afin de gérer les activités artisanales : plus de formes urbaines et plus de groupes (plus petits) à gérer.

FIGURE 11.5 – Répartition des temps de calcul par contrainte.



11.2.5 Paysage des gains lié à chaque candidat sélectionné

Les gains correspondent uniquement aux variations de coût négatives liées aux permutations évaluées (en inversant les signes). A chaque itération du processus de résolution initiale, le solveur détermine un candidat qui sera à l'origine d'une permutation gagnante. Ce graphique représente la répartition spatiale des gains

pouvant être obtenus (voir la figure 11.6) en impliquant chaque cellule de la grille à la permutation gagnante (i.e. avec le candidat retenu). La cellule la plus sombre correspond à celle qui offre le gain le plus important.

Un graphique différent est généré à chaque itération. La figure 11.7 reprend les premières itérations une à une tandis que la figure 11.8 fait apparaître les premières itérations avec un pas de 10 en 10. Sur les exemples, les représentations du début font apparaître plusieurs cellules très foncées, signe que plusieurs gains importants sont réalisables à partir d'un même candidat. Mais l'aspect foncé disparaît rapidement laissant entrevoir des gains moins significatifs. D'autre part, la répartition spatiale de ces gains varie et leur nombre également, ceci pouvant être dû à la forme urbaine associée au candidat, les contraintes spatiales spécifiques qui s'appliquent n'étant pas les mêmes d'une forme urbaine à une autre (voir la section 8.3.3).

Le nombre de cellules proposant un gain peut être important au début du processus en s'affichant avec plusieurs nuances de gris. Mais très rapidement, ce nombre se réduit et les cellules offrant un gain deviennent de plus en plus claires. Il est donc plus difficile de trouver une permutation gagnante au fur et à mesure de la résolution. Un tirage aléatoire d'une cellule gagnante pourrait être envisagé au début du processus mais ce choix ne semble pas judicieux pour les étapes suivantes.

A la fin du processus de résolution, une procédure automatique rassemble tous les graphiques générés dans une vidéo unique pour représenter l'évolution du paysage des gains au fil de la résolution.

FIGURE 11.6 – Paysage des permutations gagnantes avec le candidat retenu pour une itération donnée. Pour l'itération courante (n° 1), un candidat (i.e. cellule non représentée ici) a déjà été identifié pour participer à une permutation gagnante. Ce graphique représente la répartition spatiale des gains pouvant être obtenus en impliquant une à une chaque cellule de la grille à la permutation gagnante (i.e. avec le candidat retenu) : plus le gain potentiel est important, plus la cellule est sombre.

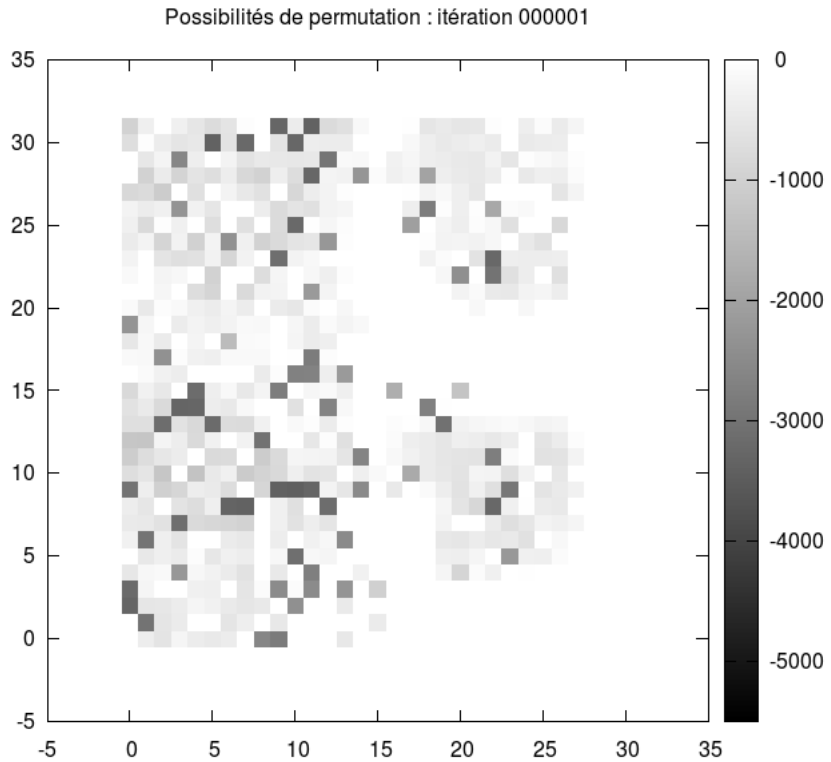


FIGURE 11.7 – Évolution du paysage des permutations gagnantes avec les candidats retenus au fil de la résolution (i.e. à chaque itération), limitée aux 12 premières itérations. Chaque graphique correspond à une itération. A chaque itération, un candidat est identifié pour participer à une permutation gagnante. Chaque graphique correspond au paysage des permutations gagnantes avec le candidat retenu pour l'itération indiquée.

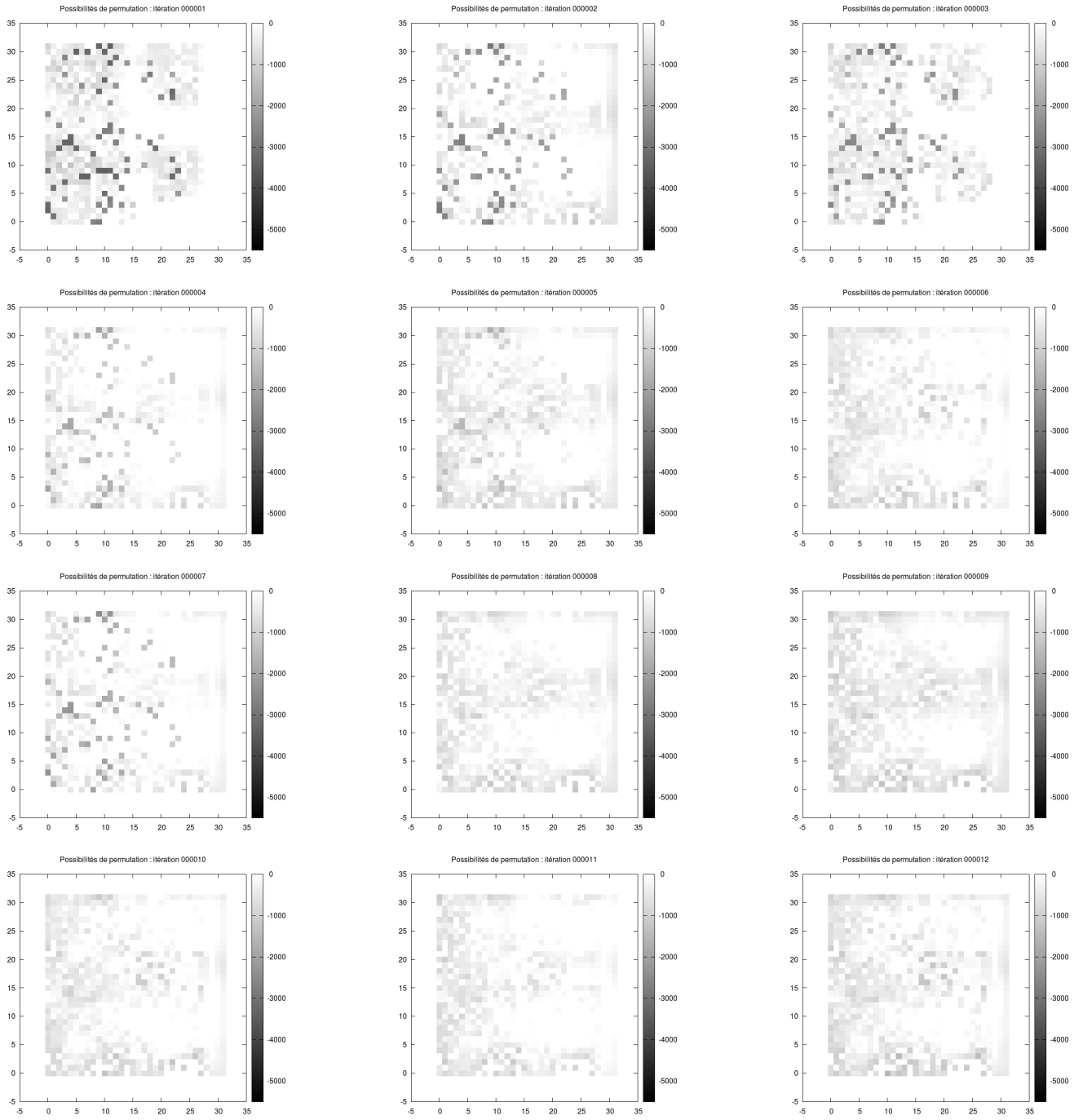
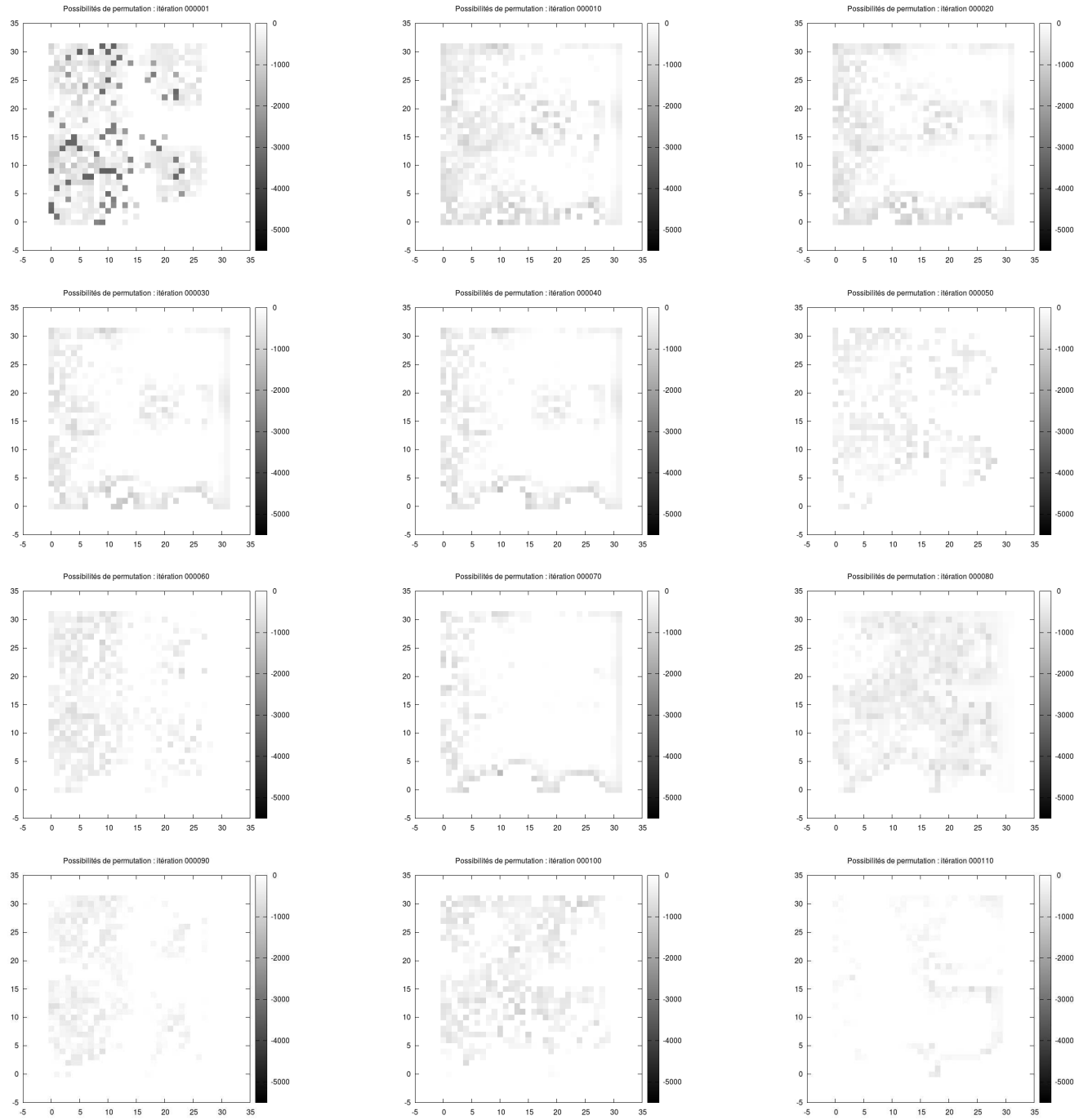


FIGURE 11.8 – Évolution du paysage des permutations gagnantes avec les candidats retenus au fil de la résolution (toutes les 10 itérations). Chaque graphique correspond à une itération. A chaque itération, un candidat est identifié pour participer à une permutation gagnante. Chaque graphique correspond au paysage des permutations gagnantes avec le candidat retenu pour l'itération indiquée.



11.2.6 Comparaison des performances entre différentes versions de l'algorithme de résolution

Comme indiqué dans la section sur le paysage des gains (voir section 11.2.5), un gain correspond à une variation de coût négative liée à une permutation gagnante (en inversant son signe). On parlera également ici de durée cumulée : il s'agit du temps mis par le processus de résolution pour atteindre une itération

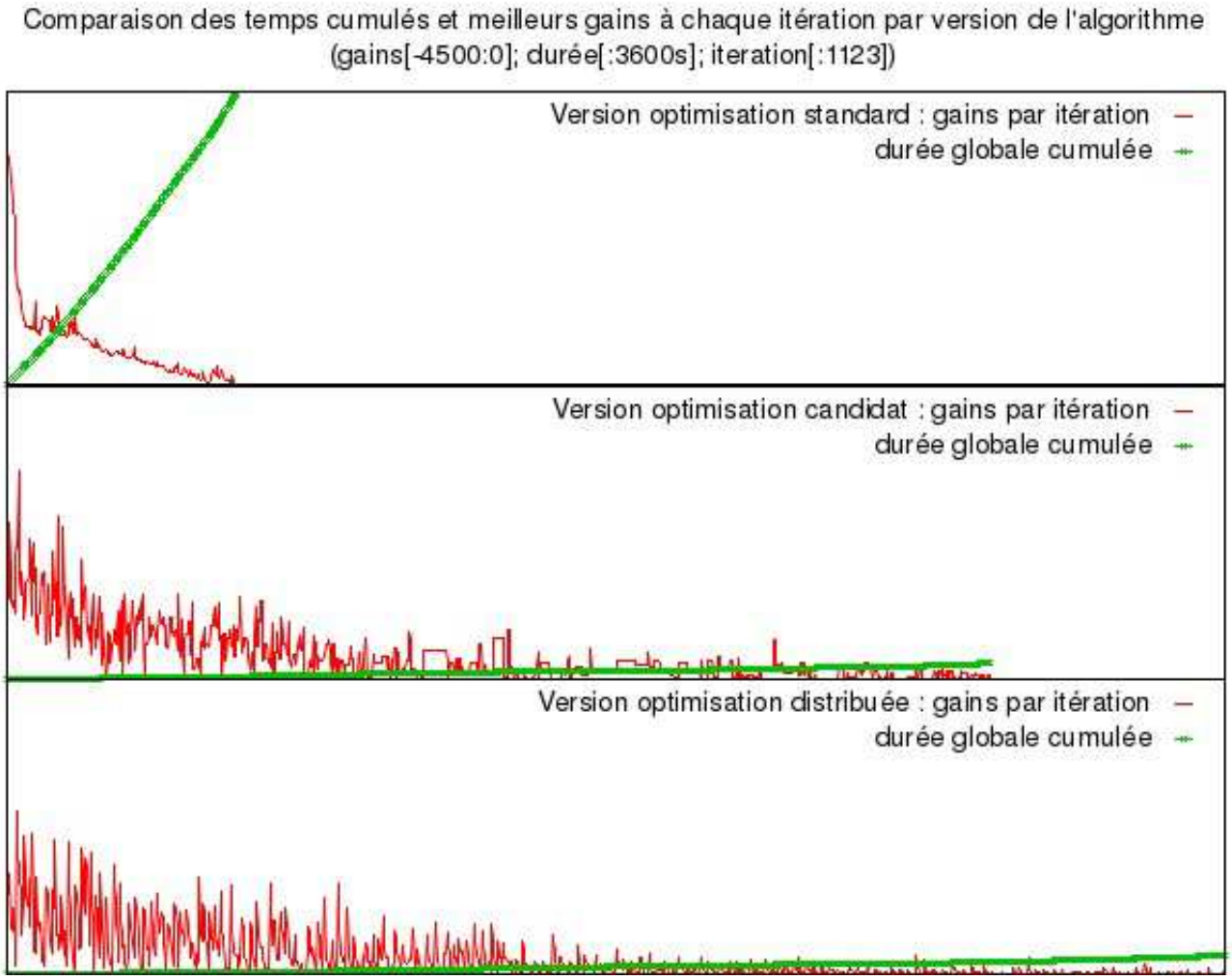
donnée. Par exemple, la première itération est atteinte avec une durée cumulée = 0, la centième itération correspond à une durée cumulée de 50 s, etc. Cette durée cumulée dépend bien évidemment de l'algorithme employé pour résoudre un problème.

Sur un même graphique, nous comparons les performances de trois versions de l'algorithme de résolution pour une même instance de problème en représentant à chaque itération :

- les gains obtenus sur le coût global de la configuration courante ;
- la durée cumulée utilisée par le processus de résolution depuis son démarrage.

En faisant apparaître ces résultats pour chaque version, on peut facilement comparer l'évolution des gains au cours des processus de résolution et le temps global nécessaire à chaque version.

FIGURE 11.9 – Comparaison des temps et des gains obtenus à chaque itération entre différentes versions de l'algorithme de résolution. L'instance correspond à une représentation simplifiée pour une grille 32x32. Les temps sont exprimés en secondes. Les échelles de temps et de gains sont identiques d'une figure à l'autre. L'axe des abscisses correspond à l'évolution du nombre des itérations au cours d'une résolution.



L'exemple proposé en figure 11.9 remonte au 14 mars 2013 et nous le reprenons ici pour illustrer cette représentation (type de graphique) qui fait partie du tableau de bord, pas pour comparer précisément les performances des différentes versions qui ont évolué depuis et que nous décrivons rapidement ci-dessous sans rentrer dans les détails. La version *standard* recherche à chaque itération la meilleure permutation à appliquer sur la configuration courante en évaluant toutes les combinaisons possibles. Il s'agit de la version de référence correspondant à une *méthode de plus forte descente* (voir la section 5.6.4). La version *candidat*

recherche la meilleure permutation à chaque itération en évaluant les permutations possibles uniquement à partir d'un meilleur candidat identifié (conforme à la méthode *Adaptive Search*, cf. section 5.6.9). La version *distribuée* correspond au protocole n° 4 (cf. section 9.4.3.3), avec un découpage de la grille en 4 partitions confiées à 10 processus esclaves. Chaque processus esclave recherche 24 permutations au maximum à partir d'un seul candidat. Le processus maître collecte alors l'ensemble des permutions provenant des différents esclaves et applique en série les meilleures permutations collectées encore capables de produire un gain.

L'exemple de la figure 11.9 montre une version standard capable de trouver les meilleurs gains à chaque itération mais dont le temps de résolution s'envole. Les deux autres versions (candidat et distribuée) nécessitent plus d'itérations, mais elles s'exécutent beaucoup plus rapidement. Ceci est lié au fait qu'elles travaillent avec un nombre limité de candidats (voir la section 9.3), ce qui réduit très fortement le nombre de permutations à évaluer à chaque itération. En contrepartie, il n'est pas sûr que la permutation retenue pour une itération soit la meilleure. Toutefois, ce schéma indique que les candidats identifiés lors d'une résolution sont pertinents pour générer des gains. Certes, les variations dans les gains trouvés sont importantes pour la version candidat, mais on atteint tout de même des niveaux de gain comparables à la version standard.

La version distribuée fait apparaître une variation des gains plus importante encore que le version candidat, ceci étant lié à son mode de fonctionnement qui permet d'appliquer plusieurs permutations en série (voir la section 9.3.4), certaines permutations retenues pouvant subir une dégradation importante par rapport aux gains prévus. Mais en compensation, son fonctionnement permet de trouver un nombre supérieur de permutations.

11.3 Expérimentations liées à la résolution initiale

11.3.1 Comparaison entre les principales versions et modes de fonctionnement

Nous comparons dans la figure 11.10 les temps de calcul pour une résolution initiale lancée dans différentes conditions :

- version mono-processus ou distribuée (uniquement avec le protocole n° 4, voir la section 9.4.3.3) ;
- mode multi-candidat activé ou non ;
- mode multi-permutation activé ou non ;
- nombre de processus esclaves (slaves) plus ou moins important.

Les tests sont réalisés sur la ville de Marne-La-Vallée qui contient 9038 cellules libres (i.e. variables). Les résolutions s'arrêtent lorsqu'elles atteignent une qualité de solution acceptable ou dès qu'un timeout spécifié est atteint. Nous définissons une solution acceptable comme une solution ayant un coût inférieur ou égal à celui de la meilleure solution trouvée pendant la résolution distribuée mettant en œuvre 36 processus esclaves (i.e. 1 260 717).

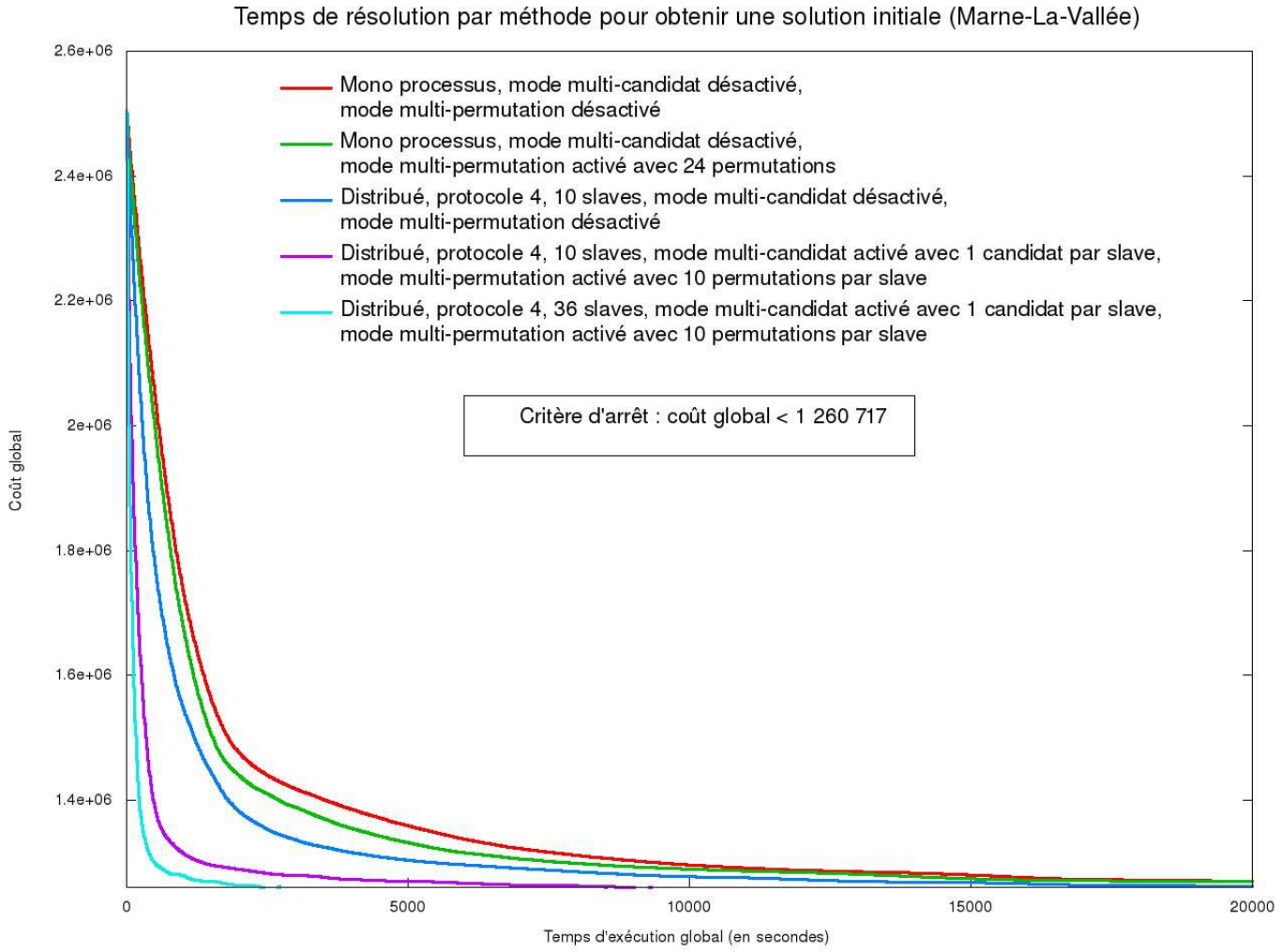
Les résultats indiqués pour les cinq expérimentations correspondent à une moyenne de 10 exécutions, chaque exécution partant d'une instance aléatoire différente. Entre chaque expérimentation, on utilise néanmoins les mêmes instances aléatoires de départ pour que les comparaisons soient cohérentes (cf. section 11.1.3.2).

Lors de ces expérimentations, nous utilisons les serveurs *granduc* situés au Luxembourg, avec 22 nœuds, chacun ayant 2 CPU Intel à 2.0 GHz, 4 cores/CPU et 15 GB RAM.

Concernant la contrainte d'accessibilité (voir la section 8.3.3.3), nous avons activé la version locale qui offre de bien meilleurs temps de calcul compte tenu de la taille réelle de nos instances (e.g. permet de réduire le temps de recherche d'une itération de 4 minutes à quelques millisecondes sur une grille 64x64).

Les résultats montrent que la version séquentielle (mono-processus) est légèrement améliorée en activant le mode multi-permutation. Mais les meilleures améliorations proviennent de la version distribuée, qui tire le maximum d'avantages des modes multi-candidat et multi-permutation. Finalement, en considérant le score cible de 1 260 717, on constate un accélération $\times 11.51$ entre la version mono-processus de

FIGURE 11.10 – Tests des performances pour différentes versions (mono-processus, distribuée) et différents modes de fonctionnement (multi-candidat et multi-permutation). Comparaison des temps de résolution (en faisant une moyenne sur 10 exécutions par méthode) pour obtenir une solution initiale dont la qualité est imposée sur l'instance de Marne-La-Vallée. La version distribuée est évaluée avec le protocole n° 4.



base (modes multi-candidat et multi-permutation désactivés) et la version distribuée (36 processus esclaves, mode multi-candidat activé avec 1 candidat par esclave, mode multi-permutation activé avec 10 permutations par esclave).

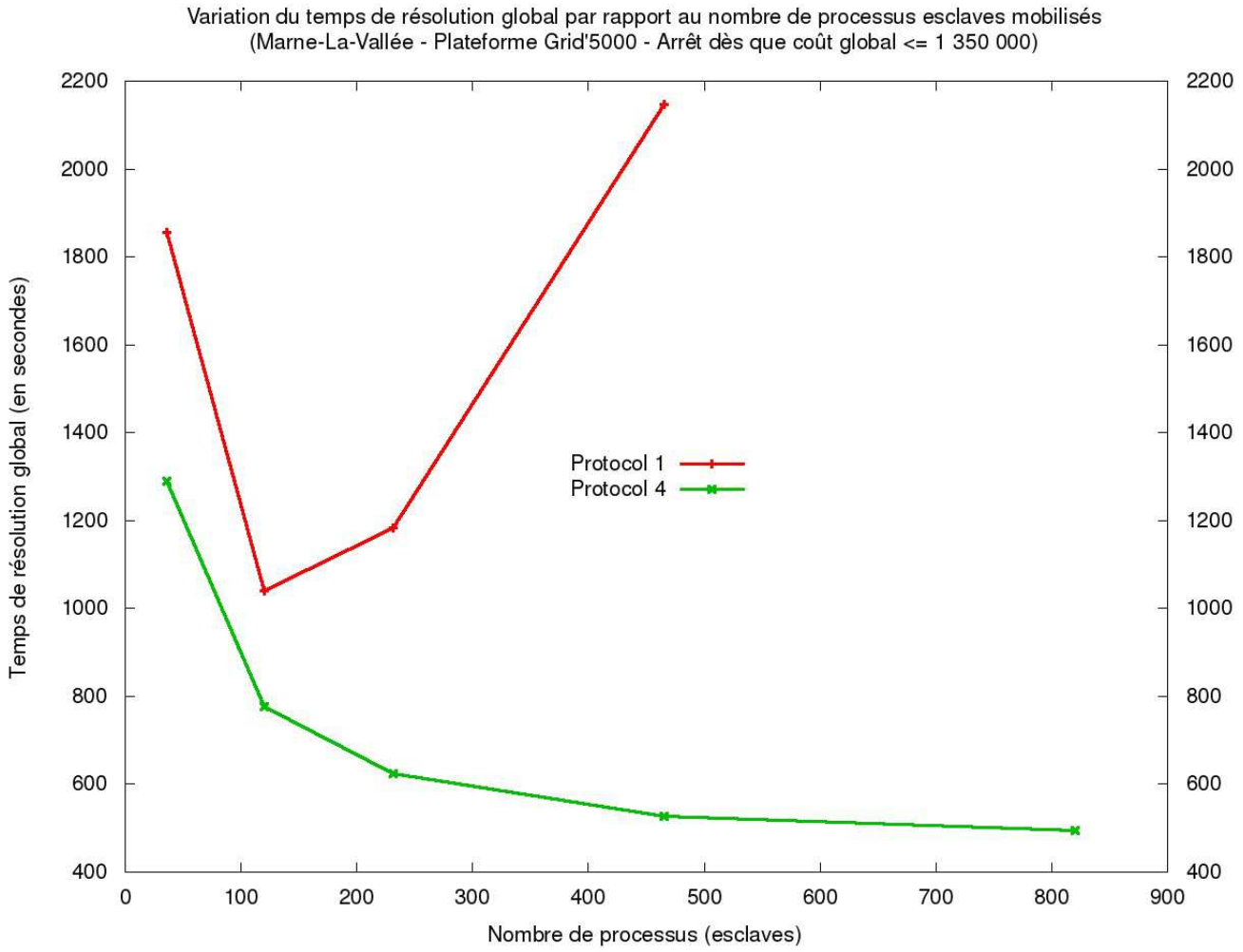
Ces premiers résultats montrent que l'on peut exécuter nos algorithmes d'optimisation sur des villes de taille réelle en moins d'une heure, ce qui paraît suffisant pour les besoins des urbanistes.

11.3.2 Évaluation des protocoles 1 et 4 de la version distribuée

Pour cette série d'expérimentations sur Grid'5000, nous avons utilisé les serveurs *griffon* de Nancy équipés de processeurs Intel Xeon L5420 cadencés à 2,5 Ghz. Chaque nœud dispose de 2 CPU et chaque CPU de 4 cores, ce qui fait 8 cores disponibles par nœud.

Concernant les protocoles 1 et 4 qui ont été évalués ici, le mode multi-candidat est activé avec 1 candidat par esclave. On récupère 10 permutations sur chaque esclave et pour rejouer une permutation sur le maître, son gain ne doit pas varier (i.e. diminution par rapport à l'évaluation initiale) de plus de 80%.

FIGURE 11.11 – Recherche du nombre optimal de processus esclaves à mobiliser.



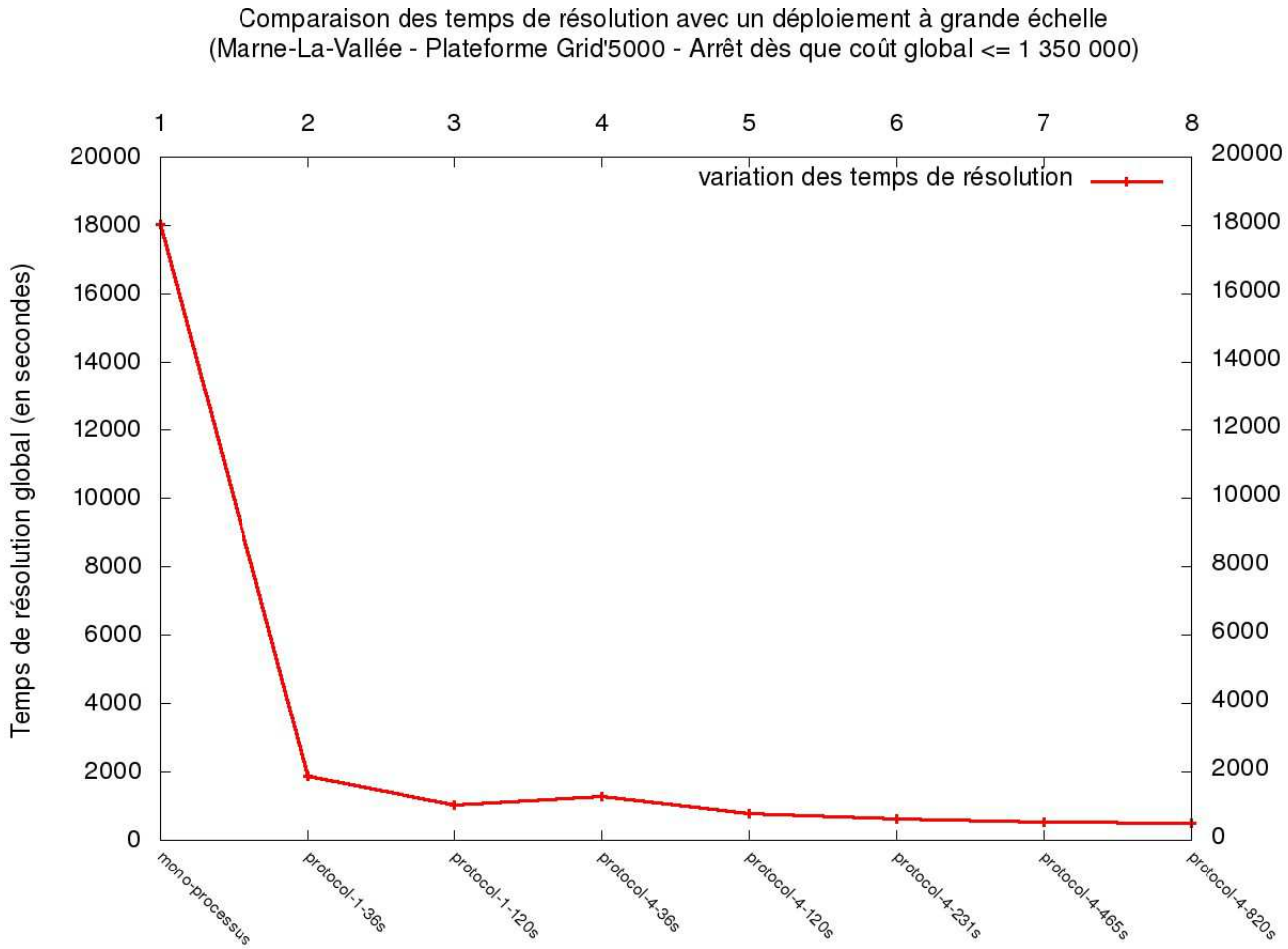
11.3.2.1 Nombre optimal de processus esclaves à mobiliser

Avec la figure 11.11, nous cherchons à déterminer le nombre optimal de processus esclaves à mobiliser sur le problème de Marne-La-Vallée afin d'obtenir les meilleurs temps de résolution possibles pour atteindre un coût global cible fixé à 1 350 000. Ce coût global cible a évolué par rapport aux expérimentations de la figure 11.10 car le paramétrage des contraintes a été modifié entre temps.

Nous avons évalué les performances pour 36, 120, 231 et 465 esclaves sur les deux protocoles. Le protocole n° 4 a également été évalué avec 820 esclaves, en réalisant toujours une moyenne des résultats sur 10 exécutions. Avec un nombre de processus esclaves identique, le protocole n° 4 propose toujours des temps de résolution inférieurs à ceux du protocole n° 1. Pour le protocole n° 1, on peut estimer un nombre optimal de processus compris entre 100 et 200 compte tenu de la granularité de nos tests. Pour le protocole n° 4, il semblerait que le nombre optimal soit situé bien au delà, aux alentours de 800 esclaves.

Ces résultats mériteraient d'être approfondis en évaluant les temps de résolution pour un nombre d'esclaves plus fin sur certaines portions de la courbe (e.g. incrémenter le nombre d'esclaves de 10 en 10 entre 100 et 200 esclaves pour le protocole n° 1). Néanmoins, ils apportent la démonstration que le protocole n° 4 est plus performant que le protocole n° 1 pour un nombre d'esclaves équivalent, et que ce protocole supporte également une bien meilleure montée en charge : 495 secondes, soit un peu plus de 8 minutes pour résoudre le problème de Marne-La-Vallée, en mobilisant 820 esclaves.

FIGURE 11.12 – Comparaison des temps de résolution entre la version mono-processus et la version distribuée déployée à grande échelle. Les protocoles 1 et 4 sont des protocoles distribués. Les intitulés « protocol-[x]-[y]s » identifie le n° du protocole (x) et le nombre de processus esclaves mobilisés (y).



11.3.2.2 Comparaison des temps de résolution entre la version mono-processus et la version distribuée déployée à grande échelle

Finalement, nous comparons avec la figure 11.12 les performances des protocoles 1 et 4 de la version distribuée avec la version mono-processus pour évaluer les accélérations maximales pouvant être obtenues en mobilisant un nombre important de processus esclaves.

Concernant la version mono-processus, le mode multi-candidat est désactivé alors que le mode multi-permutation est activé avec 10 permutations. La meilleure accélération par rapport à la version mono-processus est obtenue par le protocole distribué n° 4 en mobilisant 820 processus esclaves, on mesure alors une accélération de $\times 36.45$.

11.4 Retour des urbanistes

11.4.1 Évaluation des solutions initiales

Les urbanistes impliqués sur le projet SUSTAINS ont cherché un moyen d'analyser de manière précise la répartition des formes urbaines sur le territoire de Marne-La-Vallée, ces éléments étant automatiquement

générés par notre résolution initiale. Pour cela, ils ont sélectionné à partir de nos résultats certaines formes urbaines qu'ils ont représentées séparément. Ils ont ainsi généré différentes cartes qui permettent de visualiser la répartition des formes urbaines retenues en fonction de la zone d'intensité. Ces cartes ont été réalisées pour les formes urbaines suivantes :

- l'habitat collectif R+4 et R+7 (cf. figure 11.13) ;
- les zones d'activités industrielles et artisanales (cf. figure 11.14) ;
- les espaces verts (cf. figure 11.15).

Une dernière carte reprend ensemble différentes formes urbaines positionnées par nos procédures d'optimisation : les établissements scolaires avec les différentes formes d'habitat, ceci pour évaluer leur localisation les unes avec les autres (cf. figure 11.16).

A partir de ces différentes représentations, ils ont pu analyser nos résultats au regard de leurs règles métier.

Nous reprenons telle quelle leur conclusion :

« *Ces représentations permettent de vérifier que la forme urbaine ne génère pas d'incohérence majeure, mais également de considérer les voies possibles d'amélioration de ce module. En ce qui concerne par exemple les établissements scolaires, on peut ainsi vérifier que les lycées correspondent bien à deux îlots (une taille minimale est en effet requise afin que toutes les filières puissent y être représentées). De plus, on observe qu'un collège peut être attenant à un lycée* ».

FIGURE 11.13 – Répartition de l'habitat collectif par rapport aux niveaux d'intensité urbaine (intensités : 2, 3 et 5). Formes urbaines positionnées sur la carte par notre solveur suite à une résolution initiale. Figure extraite du rapport SUSTAINS Livrable 3.1 : EXPERIMENTATION SUR LE TERRITOIRE DE MARNE-LA-VALLEE. *Avec l'accord des auteurs :* Bernard COUTROT (SIMARIS) & Valérie GACOGNE (COMPLEXIO).

FIGURE 11.14 – Répartition des zones d'activité par rapport aux niveaux d'intensité urbaine (intensités : 2, 3 et 5). Formes urbaines positionnées sur la carte par notre solveur suite à une résolution initiale. Figure extraite du rapport SUSTAINS Livrable 3.1 : EXPERIMENTATION SUR LE TERRITOIRE DE MARNE-LA-VALLEE. *Avec l'accord des auteurs :* Bernard COUTROT (SIMARIS) & Valérie GACOGNE (COMPLEXIO).

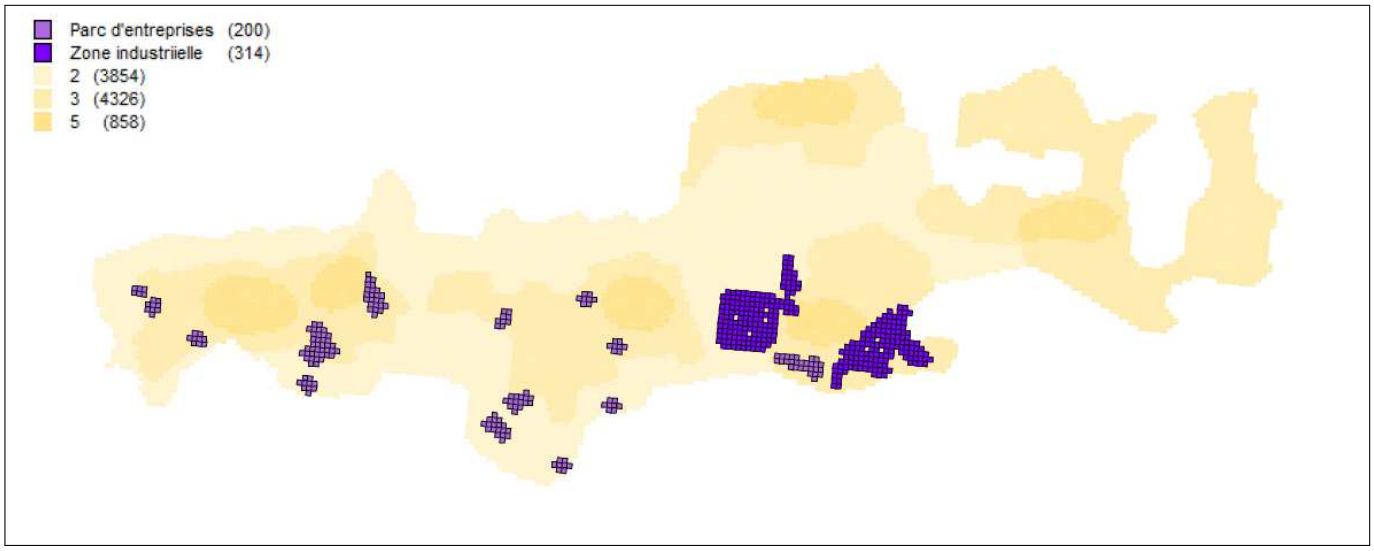


FIGURE 11.15 – Répartition des espaces verts par rapport aux niveaux d'intensité urbaine, hors espaces de respiration (intensités 2, 3 et 5). Formes urbaines positionnées sur la carte par notre solveur suite à une résolution initiale. Figure extraite du rapport SUSTAINS Livrable 3.1 : EXPERIMENTATION SUR LE TERRITOIRE DE MARNE-LA-VALLEE. *Avec l'accord des auteurs :* Bernard COUTROT (SIMARIS) & Valérie GACOGNE (COMPLEXIO).

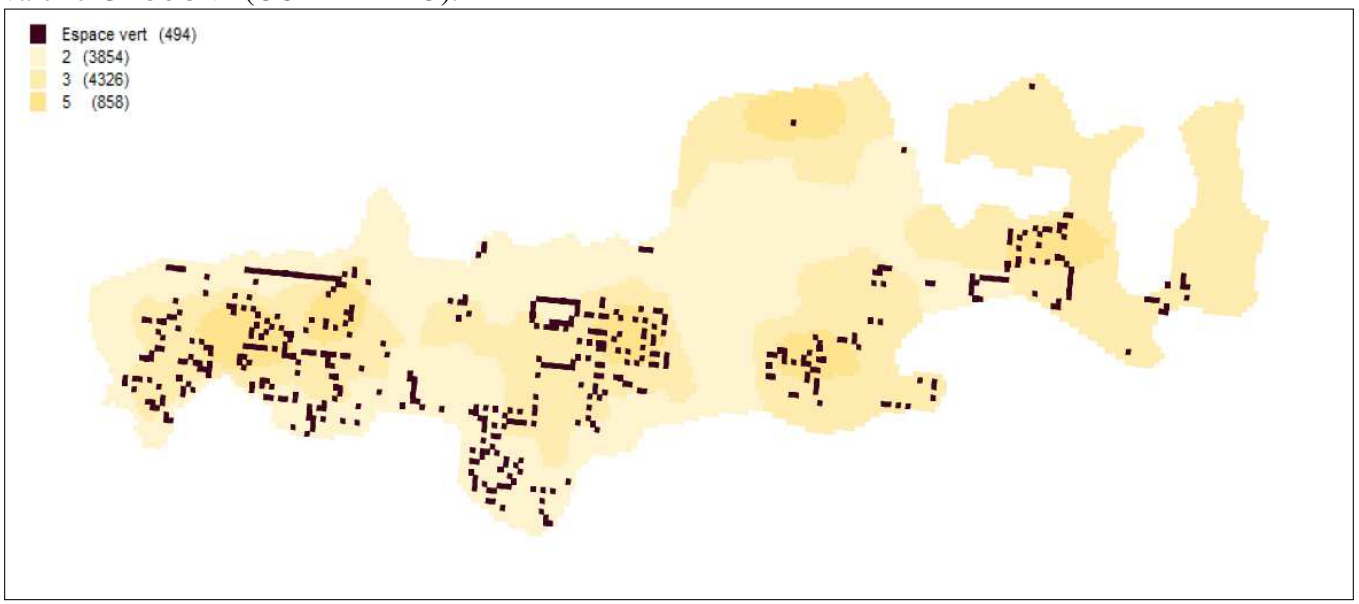
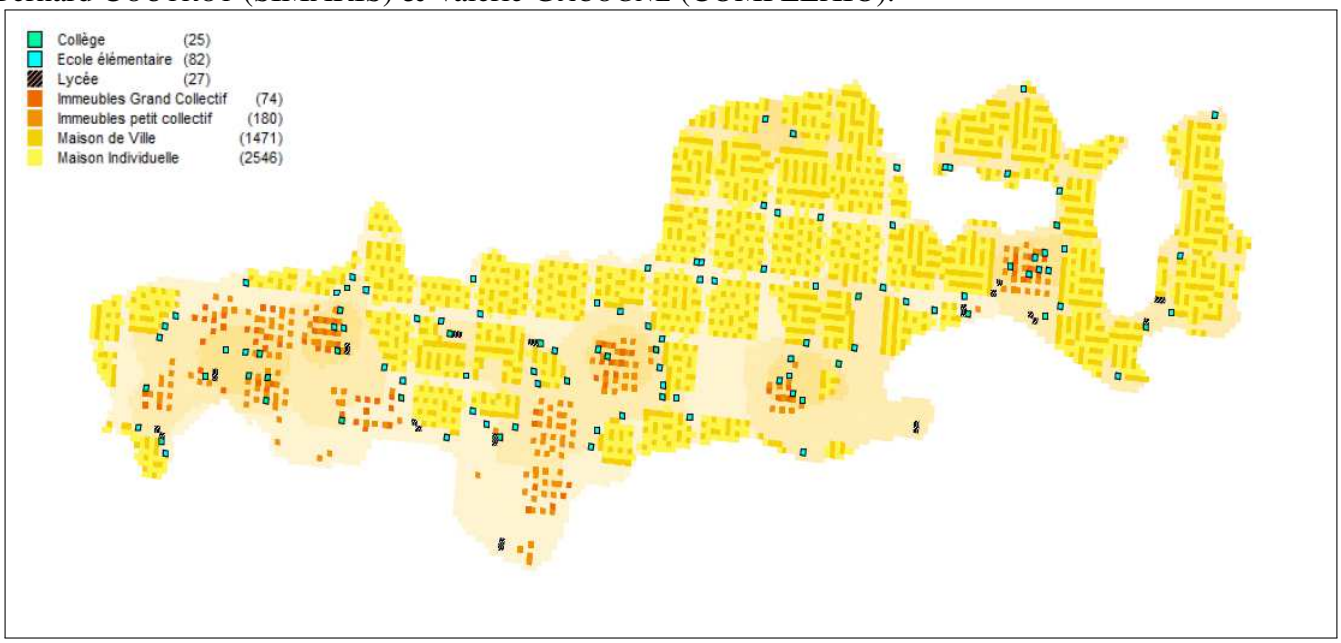


FIGURE 11.16 – Visualisation des établissements scolaires et de l'habitat. Formes urbaines positionnées sur la carte par notre solveur suite à une résolution initiale. Figure extraite du rapport SUSTAINS Livrable 3.1 : EXPERIMENTATION SUR LE TERRITOIRE DE MARNE-LA-VALLEE. *Avec l'accord des auteurs :* Bernard COUTROT (SIMARIS) & Valérie GACOGNE (COMPLEXIO).



11.4.2 Évaluation du mode interactif

Trois urbanistes expérimentés ont évalué de façon moins formelle les bénéfices du mode interactif au travers de discussions ouvertes. Au cours des démonstrations avec eux, nous avons passé *sous silence* la sélection et le chargement d'un scénario de travail, tellement cette opération parait naturelle. Nous avons donc récupéré une ville déjà organisée (résultat d'une résolution initiale) pour la modifier manuellement. Si l'on se réfère à la figure 10.2 qui met en parallèle la ville de Marne-La-Vallée organisée sous une forme aléatoire (en haut) et sous une forme optimisée (en bas), il parait beaucoup plus facile d'opérer des arrangements sur la carte optimisée qui est déjà bien structurée par rapport à l'autre totalement désorganisée. De plus, une ré-optimisation locale est beaucoup plus rapide sur une partie perturbée ayant déjà fait l'objet d'une optimisation.

Passée cette étape de chargement d'un scénario de travail, les urbanistes ont été très intéressés par la possibilité de réarranger les éléments urbains sur la ville pendant que la carte se réorganise, en maintenant à la volée la cohérence des contraintes. Dans la pratique, les manipulations des utilisateurs sont susceptibles de réduire les coûts sur la contrainte d'interaction par exemple, mais il est commode de ne pas avoir à traiter manuellement ces contraintes structurelles. Un autre exemple particulièrement expressif concerne la contrainte de regroupement qui provoque le rassemblement des activités artisanales ou industrielles autour des éléments d'un groupe fragmenté en cours de déplacement.

La carte de chaleur (voir la section 10.6.1) a été particulièrement appréciée car elle permet de repérer de façon très intuitive et naturelle les régions de la ville dont les propriétés urbaines sont les plus dégradées. Son actualisation « en temps réel » permet d'évaluer les impacts en cours de manipulation et guide les déplacements de l'utilisateur, ce qui constitue une aide efficace lors de ses choix.

Le graphique qui propose un historique d'évolution des coûts (voir la section 10.6.2) apparaît moins intuitif et a nécessité une explication préalable avant d'être accepté des utilisateurs. La partie à gauche de l'écran, reprenant le détail des coûts par contrainte pour la configuration courante, pourrait être avantageusement remplacée par un graphique en forme de *camembert* plus rapide à interpréter dans ce contexte interactif.

11.5 Conclusion

Ce chapitre commence en expliquant notre méthodologie de travail. A cette occasion, nous sommes revenus sur les premiers prototypes de notre solveur qui nous ont permis de faire des choix progressifs et éclairés. Nous avons également présenté nos différents jeux de données et l'organisation mise en place pour gérer les nombreux paramètres de configuration. Les sections implémentation et déploiement apportent des précisions sur l'architecture technique mise en œuvre, le volume des développements que nous avons réalisés en équipe et mes choix de déploiement.

La partie suivante aborde les composantes du tableau de bord proposé pour analyser le fonctionnement du solveur et le comportement des différentes contraintes. Plusieurs graphiques sont présentés : la variation des coûts, la répartition des coûts, la répartition des temps de calcul ou encore la vidéo qui séquence le paysage des gains potentiels liés à chaque candidat sélectionné. Cette partie se termine avec la présentation d'un graphique capable de comparer la proportion des gains obtenus et des temps cumulés (depuis le début des résolutions) à chaque itération pour trois versions différentes de nos algorithmes de résolution.

Nous abordons ensuite la partie « expérimentations liées à la résolution initiale ». La première section compare les performances des principales versions (mono-processus et distribuée) en exploitant des modes de fonctionnement différents (multi-candidat, multi-permutation). Nous nous focalisons ensuite sur deux protocoles distribués (1 et 4) afin d'optimiser leur fonctionnement en qualifiant un nombre de processus esclaves permettant d'obtenir les meilleures performances possibles (en temps de résolution pour un coût cible sur une instance de taille réelle). Les meilleurs temps ont été atteints grâce à un déploiement sur 820 processus esclaves avec le protocole distribué n° 4. Testé sur la ville de Marne-La-Vallée, ce protocole distribué permet une accélération $\times 36$ par rapport à la version mono-processus exploitée en mode multi-permutation activé (avec un maximum de 10 permutations).

Nous terminons ce chapitre en donnant la parole aux urbanistes qui ont pu évaluer deux aspects importants de nos travaux : (i) la qualité des solutions générées par une résolution initiale et (ii) la pertinence du mode interactif. Pour les résolutions initiales proposées, aucune incohérence majeure n'a été relevée. Des possibilités d'amélioration existent et ouvrent la voie vers des perspectives de recherche complémentaires. Le mode interactif a également suscité un fort intérêt, notamment pour ses possibilités de réarranger manuellement des éléments urbains sur une ville désormais capable de s'adapter à la volée aux changements demandés, en maintenant au mieux la cohérence des contraintes urbaines. La carte de chaleur capable de représenter les anomalies urbaines par degré d'importance et de façon spatialisée constitue une véritable aide aux utilisateurs pour opérer des choix en évaluant instantanément leurs impacts.

IV

Conclusion

12

Conclusion générale et perspectives

12.1 Conclusion générale

Nous avons commencé ce manuscrit en présentant les villes et les concepts du développement durable, deux sujets qui ont des impacts importants sur les conditions de vie des populations urbaines de plus en plus nombreuses. La conception de villes durables nécessite de prendre des décisions complexes qui engagent des moyens souvent très importants et dont les effets sont visibles sur le long terme. Un état de l'art du domaine ainsi qu'une analyse des outils mis à disposition des urbanistes nous ont permis d'étayer nos propos.

Afin d'aider les urbanistes, aménageurs et décideurs dans le processus de conception de villes durables, ma première contribution a été de proposer un modèle pour notre problème d'aménagement urbain qui repose sur des contraintes spatiales, le but étant de répartir un ensemble équilibré de formes urbaines (résidences, commerces, industries, écoles, parcs, ...) sur un territoire vierge et délimité dans l'espace tout en assurant la satisfaction des contraintes. Pour optimiser les instances de ce problème, nous avons associé à chaque contrainte spatiale une fonction de coût capable d'évaluer le niveau de violation des différentes propriétés urbaines qu'elle représente. Une description formelle de chaque contrainte a été proposée.

Ma seconde contribution porte sur la réalisation d'un solveur basé sur une méthode de recherche locale, *Adaptive Search* [47, 48], étendu avec des heuristiques novatrices pour améliorer les temps de calcul. Une version distribuée a également été proposée. Cette version distribuée a été déployée sur une grille de calcul[1] pour générer différentes solutions sur Marne-La-Vallée, une ville nouvelle qui s'étend sur 8 728 hectares et qui regroupe 234 644 habitants.

Les expérimentations menées sur cette ville ont mis en évidence des gains de temps significatifs liés à la version distribuée de notre solveur. Cette version est capable de fournir de bons résultats dans des temps raisonnables pour des villes importantes. De leur coté, les urbanistes associés au projet de recherche SUSTAINS ont vérifié la qualité des différentes solutions générées (en automatique) par notre solveur sans relever d'incohérence majeure. Ils évoquent des voies possibles pour l'amélioration de notre module.

La seconde étape du processus de résolution est interactive et elle correspond à ma troisième contribution : le concepteur peut interagir avec les solutions générées tandis que le solveur ré-optimise localement les zones autour des modifications demandées. Des retours graphiques générés à la volée permettent aux opérateurs de visualiser les impacts occasionnés par leurs modifications. Une vidéo est disponible (voir ici http://vimeo.com/80211470) pour illustrer le mode de fonctionnement interactif.

[1] https://www.grid5000.fr

12.2 Perspectives de recherche

Concernant les performances tout d'abord, la version distribuée de notre solveur peut certainement être optimisée. Pour cela, nous pourrions envisager une organisation maître-esclaves plus sophistiquée avec des agents intermédiaires permettant de limiter les communications réseau les plus pénalisantes tout en renforçant une autonomie plus grande localement (i.e. sur chaque nœud du réseau).

Suite aux évaluations des urbanistes, nous savons que le modèle que nous proposons peut être enrichi. Reste à définir avec les experts du domaine les pistes pour l'étendre ou pour l'adapter à de nouveaux besoins. En effet, le sujet que nous abordons est vaste et propice à de nouveaux projets de recherche très diversifiés. Un projet urbain peut être envisagé à différentes échelles (ville, quartier, îlot, parcelle, ...), et il traverse une succession d'étapes (pré-programmation, programmation, construction, ...) pouvant être en ruptures les unes avec les autres. Fédérer cette succession d'étapes en envisageant une continuité entre les différentes échelles constitue une perspective de recherche intéressante dans laquelle nos travaux pourraient s'intégrer.

Le développement durable repose sur un équilibre entre trois piliers de base représentés par l'économie, l'écologie et les aspects sociaux. Il pourrait être utile de regrouper les contraintes de notre modèle sur la base de ces différents piliers pour envisager une résolution multi-objectifs [51]. Cette approche renforcerait la possibilité d'arbitrer des efforts limités dans un domaine pour procurer des gains significatifs dans les autres.

Enfin, l'interaction que nous proposons est basée sur une représentation en 2D. Une interaction mettant en scène des scénarios de villes en 3D pourrait se justifier pour des projets qui s'intéressent plus spécifiquement à la perception visuelle ou aux ambiances urbaines. Aujourd'hui, rien ne nous empêche d'imaginer une immersion interactive dans une ville virtuelle animée où l'utilisateur modifierait la ville à son gré et se déplacerait dans les rues pour la percevoir sous ses différents points de vue et à différentes échelles : une expérience urbaine numérique pour améliorer la qualité des villes du futur.

Bien que notre modèle soit limité, n'adoptant qu'une vue partielle du problème, appliqué seulement à des villes nouvelles (ou à des nouveaux quartiers), il présente de premiers éléments de réponse dans la tâche complexe d'aide à la création de villes plus durables, et ouvre des perspectives encourageantes quant à la création d'une nouvelle génération d'outils de conception, dans une approche fondamentalement pluridisciplinaire.

Liste des tableaux

Table des figures

Bibliographie

[1] *Droit et politiques de renouvellement urbain*. Cahiers du GRIDAUH. : Droit de l'urbanisme. GRIDAUH, 2004. 33

[2] Emile Aarts and Jan K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997. 63, 150

[3] Mohammed Abdellaoui and Christophe Gonzales. Théorie de l'Utilité Multi-attributs. [33], pages 25–62. 59

[4] A. Adla. *Aide à la facilitation pour une prise de décision collective : proposition d'un modèle et d'un outil*. PhD thesis, Université de Toulouse III - Paul Sabatier, Spécialité : Informatique, 2010. 99

[5] Ravindra K. Ahuja, Özlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3) :75–102, 2002. 78

[6] W. Alonso. *Location and land use : toward a general theory of land rent*. Publication of the Joint Center for Urban Studies. Harvard University Press, 1964. 111, 112, 224

[7] Jean-Philippe Antoni. L'ambition de modéliser la ville. In Gérard Brun, editor, *Ville et mobilité - nouveaux regards*, pages 227–238. Economica, 2013. 128

[8] J.P. Antoni. *Modéliser la ville : Formes urbaines et politiques de transport*. Collection "Méthodes et approches". Economica, 2010. 114, 115, 132, 231

[9] Krzysztof Apt. *Principles of Constraint Programming*. Cambridge University Press, New York, NY, USA, 2003. 63, 149, 194

[10] Alejandro Arbelaez and Philippe Codognet. A GPU Implementation of Parallel Constraint-based Local Search. *Techniques foR Implementing Constraint programming Systems (TRICS)*, 2013. 87, 88, 150, 159

[11] François Ascher. Effet de serre, changement climatique et capitalisme cleantech. *Esprit*, Février :150–164, 2008. 43

[12] Olivier Bailleux. Cours de programmation par contraintes, 2011. Master-II BD-IA & Image-IA. 56

[13] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 46 :316–329, 1996. 62

[14] Nicolas Barnier. *Application de la programmation par contraintes à des problèmes de gestion du trafic aérien*. PhD thesis, Thèse doctorat informatique de l'INP Toulouse, 2002. 56

[15] Richard S. Barr, Bruce L. Golden, James P. Kelly, G. C Mauricio, William Resende, and R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1 :9–32, 1995. 75

[16] Roman Bartak. Constraint programming : In pursuit of the holy grail. In *in Proceedings of WDS99 (invited lecture*, pages 555–564, 1999. 56, 63, 149, 194

[17] R. Battiti, G. Tecchiolli, and Others. The Reactive Tabu Search. *ORSA Journal on Computing*, 6 :126, 1994. 127, 128, 221

[18] Roberto Battiti and Marco Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29(4) :610–637, 2001. 75

[19] G. Bauer and J.M. Roux. *La rurbanisation ; ou, La ville éparpillée*. Collection Espacements. Éditions du Seuil, 1976. 25

[20] François BELLANGER. La recherche urbaine à l'heure de la ville 2.0, Oct 2012. Entretien scientifique. 36, 40

[21] A. Bellicha. Maintenance of Solution in a Dynamic Constraint Satisfaction Problem. In *Proc. of Applications of Artificial Intelligence in Engineering VIII*, pages 261–274, Toulouse, France, 1993. 107

[22] Amit Bellicha, Bertrand Neveu, Brigitte Trousse, Christian Bessière, Christine Gaspin, David Lesaint, Fabrice Bouquet, Florence Dupin de Saint-Cyr, Gilles Trombettoni, Gérard Verfaillie, Hélène Fargier, Jean-Charles Régin, Jean-Pierre Rellier, Jérôme Amilhastre, Jérôme Lang, Khaled Ghedira, Marie-Catherine Vilarem, Martin Cooper, Mouhssine Bouzoubaa, Olivier Lhomme, Philippe Charman, Philippe David, Philippe Janssen, Philippe Jégou, Pierre Berlandier, Roger Martin-Clouaire, Thomas Schiex, and Tibor Kökény. Autour Du Problème De Satisfaction De Contraintes. In *In Actes des 5èmes journées nationales du PRC GDR Intelligence Artificielle*, pages 159–178, 1994. 106

[23] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957. 63

[24] Sahbi BEN ISMAIL. Cours d'ingénieurs généralistes - Introduction à l'optimisation combinatoire, 2012. 56, 221

[25] S. Ben-Mena. Introduction aux méthodes multicritères d'aides à la décision. *Biotechnologie, Agronomie, Société et Environnement*, 2000. 59

[26] F. Benhamou. Stratégie de développement de la recherche de l'université de nantes, programme université 2020, 2014. 30

[27] Nicolas Berger. *Modélisation et résolution en programmation par contraintes de problèmes mixtes continu/discret de satisfaction de contraintes et d'optimisation*. PhD thesis, Université de Nantes, October 2010. 58

[28] Antoine Billot and Jacques-François Thisse. Modèles de choix individuels discrets : théorie et applications à la micro-économie. *Revue économique*, 46(3) :921–931, 1995. 117

[29] James R. Bitner and Edward M. Reingold. Backtrack programming techniques. *Commun. ACM*, 18(11) :651–656, November 1975. 68

[30] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3) :268–308, September 2003. 56, 72, 81, 82, 223

[31] Stefan Bock and Otto Rosenberg. A new parallel breadth first tabu search technique for solving production planning problems. *International Transactions in Operational Research*, 7(6) :625 – 635, 2000. 94

[32] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'89, pages 979–984, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. 104

[33] Denis Bouyssou, Didier Dubois, Marc Pirlot, and Henri Prade. *Concepts et méthodes pour l'aide à la décision, analyse multicritère*, volume 3. Hermés, 2006. 59, 229

[34] Jurgen Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. 60

[35] R. Braune and G. Zäpfel. *Metaheuristic Search Concepts : A Tutorial with Applications to Production and Logistics*. Springer, 2010. 80, 223

[36] Marilyn A. Brown, Frank Southworth, and Andrea Sarzynski. The geography of metropolitan carbon footprints. *Policy and Society*, 27(4) :285 – 304, 2009. 43

[37] E.W. Burgess. *The Growth of the City : an Introduction to a Research Project*. University of Chicago Press, 1925. 21, 112, 224

[38] S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo : A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3) :357–380, May 2004. 97

[39] Aurore Cambien. La modélisation urbaine : une approche historique. [8], pages 19 – 49. 114, 115

[40] Yves Caniou, Philippe Codognet, Daniel Diaz, and Salvador Abreu. Experiments in parallel constraint-based local search. In *EvoCOP'11, 11th European Conference on Evolutionary Computation in Combinatorial Optimisation*, Lecture Notes in Computer Science, Torino, Italy, 2011. Springer Verlag. 159

[41] Michel Cantal-Dupart. Vivre un grand paris, Oct 2013. Note ouverte au Président de la République sur le Grand Paris. 19, 34, 37

[42] V Cerny. Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1) :41–51, 1985. 65, 83, 150

[43] F. Choay. *L'Urbanisme : utopies et réalités*. Points (Paris). Éditions du Seuil, 1965. 22

[44] Solver Choco. Présentation - tutorial et documentation, 2011. 56

[45] Walter Christaller. *Central Places in Southern Germany*. Prentice-Hall, 1966. 110, 111, 224

[46] Maurice Clerc and Patrick Siarry. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires, 2004. J3eA. 59

[47] Philippe Codognet and Daniel Diaz. Yet Another Local Search Method for Constraint Solving. In *proceedings of SAGA'01*, volume 2264 of *Lecture Notes in Computer Science*, pages 73–90. Springer Verlag, 2001. 56, 66, 86, 150, 152, 219

[48] Philippe Codognet and Daniel Diaz. An Efficient Library for Solving CSP with Local Search. In *5th international Conference on Metaheuristics*, pages 1–6, Kyoto, Japan, 2003. 86, 150, 219

[49] Philippe Codognet, Daniel Diaz, and Charlotte Truchet. The Adaptive Search Method for Constraint Solving and its application to musical CSPs. In *International Workshop on Heuristics*, page 9 pages, Beijing, Chine, 2002. 87

[50] S. G. Cohen and D. E. Bailey. What makes teams work : Group effectiveness research from the shop floor to the executive suite. *Journal of Management*, 23(3) :239–291, 1997. 102

[51] Jared L. Cohon. *Multiobjective Programming and Planning*. Mathematics in science and engineering. Academic Press, 1978. 59, 220

[52] N. E. Collins, R.W. Eglese, and Golden B.L. Simulated Annealing : an Annotated Bibliography. *American Journal of Mathematical and Management Sciences*, 8(3-4) :209–307, January 1988. 84

[53] Clayton W. Commander. A survey of the quadratic assignment problem, with applications. *Morehead Electronic Journal of Applicable Mathematics*, 4 :MATH–2005–01, 2005. 126

[54] David T. Connolly. An improved annealing scheme for the {QAP}. *European Journal of Operational Research*, 46(1) :93 – 100, 1990. 127, 221

[55] L. Corbusier and J. Giraudoux. *La charte d'Athènes*. Cahiers forces vives. Éditions de Minuit, 1957. 11, 24

[56] J.-C. COURBON, D. DUBOIS, and B. ROY. Autour de l'aide à la décision et l'intelligence artificielle. IBP-Laforia, 1994. Rapport de Recherche. 102, 103

[57] Teodor Gabriel Crainic and Michel Toulouse. Parallel Strategies for Meta-heuristics. [117], pages 475–513. 76

[58] Teodor Gabriel Crainic and Michel Toulouse. Parallel Meta-Heuristics. In *Handbook of metaheuristics*, pages 497–541. Springer US, 2010. 91, 92, 93, 94, 159, 224

[59] Andrew Crooks, Christian Castle, and Michael Batty. Key challenges in agent-based modelling for geo-spatial simulation. *Computers, Environment and Urban Systems*, 32(6) :417 – 430, 2008. GeoComputation : Modeling with spatial agents. 115

[60] John Current, Mark Daskin, and David Schilling. Discrete Network Location Models. [80], pages 81–118. 125

[61] Institut d'aménagement et d'urbanisme de la région d'Île-de France. *ULTISIM, vers un modèle intégré transport-urbanisme européen : première phase : rapport final*. IAU Île-de-France, 2011. 115, 116, 117, 224

[62] G. B. Dantzig. *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*, chapter XXI. Wiley, New York, 1951. 61

[63] D. Dasgupta and Z. Michalewicz. *Evolutionary Algorithms in Engineering Applications*. U.S. Government Printing Office, 1997. 74

[64] C. de Portzamparc. *la Ville âge III*. Éditions du Pavillon de l'Arsenal, 1997. 26

[65] J. de Rosnay. *Le Macroscope : vers une vision globale*. Collection Civilisation. Éditions du Seuil, 1975. 114

[66] D. de Werra and A. Hertz. Tabu search techniques. *Operations-Research-Spektrum*, 11(3) :131–141, 1989. 85, 157

[67] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 49–54. AAAI Press/MIT Press, August 1988. 104

[68] Rina Dechter and Avi Dechter. Belief Maintenance in Dynamic Constraint Networks. *Proceedings of AAAI*, pages 37–42, 1988. 60

[69] Philippe Dehan, Enseignant-Chercheur. Cours ingénieur - modes opératoires de l'urbanisme contemporain, 2011. 19, 22, 33, 223

[70] Xavier Desjardins. Pour l'atténuation du changement climatique, quelle est la contribution possible de l'aménagement du territoire ?, March 2011. Cybergeo : European Journal of Geography [En ligne], Aménagement, Urbanisme, document 523. 43

[71] Xavier Desjardins. Formes urbaines. [88], pages 108–109. 31

[72] I. Devarenne, A. Caminada, and H. Mabed. Analysis of adaptive local search for the graph coloring problem. In *Proceedings of the 6th Metaheuristics International Conference*, MIC 2005, Vienne Autriche, 2005. 86

[73] Isabelle Devarenne, Hakim Mabed, and Alexandre Caminada. Intelligent neighborhood exploration in local search heuristics. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '06, pages 144–150, Washington, DC, USA, 2006. IEEE Computer Society. 86

[74] Luca Di Gaspero and Andrea Schaerf. Easylocal++ : An object-oriented framework for the flexible design of local-search algorithms. *Softw. Pract. Exper.*, 33(8) :733–765, July 2003. 96, 194, 197

[75] Daniel Diaz, Salvador Abreu, and Philippe Codognet. Targeting the cell/be for constraint-based local search. *Concurrency and Computation : Practice and Experience*, 24(6) :647–660, 2012. 159

[76] Mohammad Dib. *Tabu-NG : hybridation de programmation par contraintes et recherche locale pour la résolution de CSP*. PhD thesis, Université de Technologie de Belfort-Montbeliard, December 2010. 56, 66, 67, 89

[77] J. Donzelot, C. Mével, and A. Wyvekens. *Faire société : la politique de la ville aux États-Unis et en France*. La couleur des idées. Seuil, 2003. 20

[78] Marco Dorigo. Ant colony optimization : A new meta-heuristic. In *Proceedings of the Congress on Evolutionary Computation*, pages 1470–1477. IEEE Press, 1999. 55, 65

[79] J. Dréo. *Metaheuristics for Hard Optimization : Methods and Case Studies*. Springer, 2006. 73

[80] Z. Drezner and H.W. Hamacher. *Facility Location : Applications and Theory*. Springer series in operations research. Springer, 2004. 124, 149, 232, 241

[81] Alexis Drogoul. *De la simulation multi-agent à la résolution collective de problèmes*. PhD thesis, Université Paris VI, 1993. 55

[82] Dominique Dron. Origine du développment durable. [88], pages 24–25. 29, 30

[83] Claude Duvallet. *Des systèmes d'aide à la décision temps réel et distribués : modélisation par agents*. PhD thesis, Doctorat Informatique de l'Université du Havre, 2001. 99, 103, 104

[84] D. Duvivier, Ph. Preux, and E.-G. Talbi. Climbing up NP-hard hills. In *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 574–583. Springer Berlin Heidelberg, 1996. 79

[85] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1 :127–136, 1971. 70, 149

[86] A. Elkhyari. *Outils d'aide à la décision pour des problèmes d'ordonnancement dynamiques*. 2003. 107

[87] Agathe Euzen. À propos du développment durable. [88], page 23. 29, 30

[88] Agathe Euzen, Laurence Eymard, and Françoise Gaill. *Le développement durable à découvert*. A découvert. CNRS Editions, 2013. 233, 234, 235, 238, 240, 242, 244

[89] M. Fabriès-Verfaillie, P. Stragiotti, and A. Jouve. *La France des villes : le temps des métropoles ?* Bréal, 2000. 111, 224

[90] R.Z. Farahani and M. Hekmatfar. *Facility Location : Concepts, Models, Algorithms and Case Studies*. Contributions to Management Science. Physica, 2009. 125, 126, 224

[91] Thomas A. Feo and Mauricio G.C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6 :109–133, 1995. 89

[92] J. Ferber. *Les systèmes multi-agents : Vers une intelligence collective*. I.I.A. Informatique intelligence artificielle. InterEditions, 1995. 55

[93] Andreas Fink, Stefan Voß, and DavidL. Woodruff. Building reusable software components for heuristic search. 1998. 97

[94] Charles Fleurent and Jacques A. Ferland. Genetic Hybrids for the Quadratic Assignment Problem. In *DIMACS Series in Mathematics and Theoretical Computer Science*, pages 173–187. American Mathematical Society, 1993. 127, 221

[95] R. Fondi and P. Baillet. *La Révolution organiciste : entretien sur les nouveaux courants scientifiques*. Livre-club du Labyrinthe. Labyrinthe, 1986. 114

[96] J.W. Forrester. *Urban dynamics*. M.I.T. Press, 1969. 114

[97] T. Frühwirth and S. Abdennadher. *Essentials of Constraint Programming*. Cognitive Technologies. Springer, 2003. 63, 149, 194

[98] Alexandre Gachet. A framework for developing distributed cooperative decision support systems - inception phase, 2001. Informing Science - Challenges to Informing Clients : A Transdisciplinary Approach. 103

[99] Philippe Galinier and Jin-Kao Hao. A General Approach for Constraint Solving by Local Search. In *In CP-AI-OR'00*, 2000. 87, 154

[100] L. M. Gambardella, E. Taillard, and M. Dorigo. Ant colonies for the Quadratic Assignment Problem. *Journal of the Operational Research Society*, (50) :167–176, 1999. 127, 221

[101] Bruno-Laurent Garcia, Jean-Yves Potvin, and Jean-Marc Rousseau. A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints. *Computers & Operations Research*, 21(9) :1025 – 1033, 1994. 92

[102] V. Gardeux. *Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. Application à l'analyse de données de puces à ADN*. PhD thesis, Paris-Est University, Créteil, France, November 2011. 91

[103] F. Gardi, T. Benoist, J. Darlay, B. Estellon, and R. Megel. *Mathematical Programming Solver Based on Local Search*. Wiley, 2014. 97

[104] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. 58

[105] M. Gen and R. Cheng. A survey of penalty techniques in genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 804–809, May 1996. 74

[106] M. Gendreau. An Introduction to Tabu Search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, chapter 2, pages 37–54. Kluwer Academic Publishers, 2003. 86

[107] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10) :1276–1290, 1994. 86

[108] Commission Européenne. Direction générale de la politique régionale. *Les villes de demain : Défis, visions et perspectives*. Office des publications de l'Union européenne, 2011. 40

[109] Cynthia Ghorra-Gobin. Le territoire métropolitain à l'heure de la mondialisation. [88], pages 28–29. 31

[110] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, 9(6) :849–859, November 1961. 62

[111] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5) :533–549, May 1986. 65, 72, 84, 150, 155, 157, 194

[112] Fred Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3) :190–206, 1989. 65, 81, 86, 150, 155

[113] Fred Glover. Tabu Search - Part II. *ORSA Journal on Computing*, 2 :4–32, 1990. 65, 81, 86, 150, 155

[114] Fred Glover. Scatter search and path relinking. *New ideas in optimization*, pages 297–316, 1999. 65

[115] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. 56, 65, 81, 150, 155

[116] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *CONTROL AND CYBERNETICS*, 39 :653–684, 2000. 65

[117] Fred W. Glover and Gary A. Kochenberger. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer, 1 edition, 2003. 232, 238

[118] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. 65

[119] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64 :275–278, 1958. 62

[120] Yves Grafmeyer and Isaac Joseph. *L'École de Chicago. Naissance de l'écologie urbaine*. Aubier, 1984. 112

[121] S. A. Groenewegen, R. M. Smelik, K. J. de Kraker, and R. Bidarra. Procedural city layout generation based on urban land use models. In *EUROGRAPHICS*, 2009. 137

[122] William Grossin. Berger gaston, phénoménologie du temps et prospective. *Revue française de sociologie*, 6(3) :388–389, 1965. 128

[123] P. Haggett and R.J. Chorley. *Models, paradigms and the new geography*. The Trinity Press, 1967. 109

[124] S. L. Hakimi. Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph. *Operations Research*, 12(3) :450–459, 1964. 125

[125] P. Hansen. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. In *Proceedings of the Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986. 65, 84, 150, 155

[126] Pierre Hansen and Nenad Mladenovic. Variable neighborhood search : Principles and applications. *European Journal of Operational Research*, 130(3) :449–467, 2001. 89

[127] J K Hao, P Galinier, and M Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 13(2) :283–324, 1999. 56, 63, 65, 86, 89, 106, 150

[128] Jin-Kao Hao and Christine Solnon. *Algorithmes pour l'intelligence artificielle*, chapter Métaheuristiques et intelligence artificielle. Cépaduès, December 2014. 81

[129] C.D. Harris and E.L. Ullman. *The Nature of Cities*, volume 242. 1945. 112, 113, 224

[130] G. Hégron, L. Tapadinhas, and Groupement d'intérêt scientifique Modélisation urbaine (France). *Modélisation urbaine : de la représentation au projet : [actes du colloque tenu à Paris, École des Ponts-Paris Tech, 23-24 février 2011]*. RéférenceS / Commissariat général au développement durable. Commissariat général au développement durable, 2012. 35

[131] A. Hertz and D. de Werra. The tabu search metaheuristic : How we used it. *Annals of Mathematics and Artificial Intelligence*, 1(1-4) :111–121, 1990. 85, 157

[132] A. Hertz and M. Widmer. La méthode TABOU appliquée aux problèmes d'ordonnancement. *Journal européen des systémes automatisés : JESA.*, 1995. 79

[133] John H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992. 55, 65

[134] S. Holtzman. *Intelligent Decision Systems*. Teknowledge Series in Knowledge Engineering. Addison-Wesley, 1989. 99

[135] H.H. Hoos and T. Stützle. *Stochastic Local Search : Foundations & Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier Science, 2004. 82

[136] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *In Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, pages 429–444, 1987. 104

[137] H. Hoyt. *The Structure and Growth of Residential Neighborhoods in American Cities*. Federal housing administration. U.S. Government Printing Office, 1939. 112, 113, 224

[138] B. A. Huberman, R. M. Lukose, and T. Hogg. An economic approach to hard computational problems. *Science*, 27 :51–53, 1997. 94, 159

[139] D. L. Huff. Defining and Estimating a Trading Area. *Journal of Marketing*, 28,(3) :34–38, 1964. 113

[140] J.L. Huot. *La Ville neuve : une idée de l'Antiquité ?* Cahiers du G.S. 72, Terrains et théories en archéologie. Éditions Errance, 1988. 19

[141] IBM France. *Réinventons la ville. Regards sur des villes intelligentes*, 2010. Think. 19

[142] J. Jacobs. *The economy of cities*. Vintage international. Random House, 1969. 20

[143] Arun Jagota and LauraA. Sanchis. Adaptive, restart, randomized greedy heuristics for maximum clique. *Journal of Heuristics*, 7(6) :565–585, 2001. 71, 150

[144] Yaochu Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *Trans. Evol. Comp*, 9(3) :303–317, June 2005. 60

[145] EllisL. Johnson. Modeling and strong linear programs for mixed integer programming. In SteinW. Wallace, editor, *Algorithms and Model Formulations in Mathematical Programming*, volume 51 of *NATO ASI Series*, pages 1–43. Springer Berlin Heidelberg, 1989. 63

[146] Laetitia Jourdan. *Métaheuristiques Coopératives : du déterministe au stochastique*. Hdr, Université des Sciences et Technologie de Lille - Lille I, September 2010. 89, 95, 98, 224

[147] Narendra Jussien, Guillaume Rochart, and Xavier Lorca. Choco : an Open Source Java Constraint Programming Library. In *CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08)*, pages 1–10, Paris, France, France, 2008. 149, 194

[148] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM. 61

[149] P. G. W. Keen and M. S. Scott Morton. Decision Support Systems : An Organizational Perspective. Addison-Wesley, Reading, 1978. 102, 177

[150] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995. 55, 65

[151] Gregory E. Kersten and Geoffrey R. Mallory. Supporting problem representations in decisions with strategic interactions. *European Journal of Operational Research*, 46(2) :200 – 215, 1990. 102

[152] Madjid Khichane. *Optimisation sous contraintes par Intelligence Collective Auto-adaptative*. Thèse de doctorat en informatique, Université Claude Bernard, October 2010. 89

[153] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598 :671–680, 1983. 65, 83, 150

[154] A. Koestler and G. Fradier. *Le Cheval dans la locomotive : le Paradoxe humain*. Génie et folie de l'homme. Calmann-Lévy, 1968. 114

[155] Tjalling C. Koopmans and Martin Beckmann. Assignment Problems and the Location of Economic Activities. *Econometrica*, 25(1) :53–76, 1957. 125

[156] Jakob Krarup and Peter Mark Pruzan. The simple plant location problem : Survey and synthesis. *European Journal of Operational Research*, 12(1) :36 – 81, 1983. 125

[157] Krzysztof Krawiec. Metaheuristic Design Pattern : Candidate Solution Repair. In *Proceeding of the sixteenth annual conference on Genetic and evolutionary computation conference*, GECCO '14, New York, NY, USA, 2014. ACM. 74

[158] M. Kwartler and R Bernard. Communityviz : an integrated planning support system. *Planning Support Systems*, November 2001. 49

[159] Thomas J. Laffey, Preston A. Cox, James L. Schmidt, Simon M. Kao, and Jackson Y. Read. Real-time knowledge-based systems. *AI Magazine*, 9(1) :27–45, 1988. 103

[160] Yannick Lageat. Variation du trait de côte. [88], pages 76–77. 29

[161] Manuel Laguna, Rafael Marti, and Vicente Campos. Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Computers & Operations Research*, 26(12) :1217 – 1230, 1999. 81

[162] Tony Lambert. *Hybridation de méthodes complètes et incomplètes pour la résolution de CSP*. These, Université de Nantes, October 2006. 89

[163] A. H. Land and A. G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3) :497–520, 1960. 62

[164] Pedro Larraanaga and Jose A. Lozano. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. 70

[165] Catherine Larrère. Éthique et philosophie de l'environnement. [88], pages 48–49. 30

[166] P. Lavedan. *Géographie des villes*. Géographie humaine. Gallimard, 1936. 19

[167] J.L. Le Moigne. *Les systèmes de décision dans les organisations*. Systèmes décisions. Section 1 : Systèmes, plans, contrôles. Presses universitaires de France, 1974. 100, 101, 224

[168] Julien Lepagnot. *Conception de métaheuristiques pour l'optimisation dynamique : application à l'analyse de séquences d'images IRM*. These, Université Paris-Est, December 2011. 60

[169] Arnaud Letort. *Passage à l'échelle pour les contraintes d'ordonnancement multi-ressources*. These, Ecole des Mines de Nantes, October 2013. 63

[170] P. Lévine, G. Benchimol, and J.C. Pomerol. *Systèmes experts dans l'entreprise*. Traité des Nouvelles Technologies : Série décision assistée par ordinateur. Hermes, 1990. 101

[171] P. Lévine and J.C. Pomerol. *Systèmes interactifs d'aide à la décision et systèmes experts*. Hermes, 1989. 100, 101, 102, 103, 177

[172] L. Lobjois. *Problèmes d'optimisation combinatoire sous contraintes : vers la conception automatique de méthodes de résolution adaptées à chaque instance*. 1999. 103, 105, 106, 224

[173] Pierre-Yves Longaretti. Changements globaux. [88], pages 40–41. 29

[174] Michel Loreau. Modélisation et expérimentation pour la biodiversité. [88], pages 90–91. 30

[175] A. Lösch. *Die räumliche Ordnung der Wirtschaft*. G. Fischer, 1940. (L'organisation spatiale de l'économie - ouvrage non traduit en français). 111

[176] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. A Beginner's Introduction to Iterated Local Search. In *Proceedings of MIC 2001*, Porto, Portuga, July 2001. 83

[177] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated Local Search. [117], pages 321–353. 65, 83, 150

[178] F. Loyer and H. Guéné. *Henri Sauvage : les immeubles à gradins*. Mardaga, 1987. 23

[179] A.K. MACKWORTH. Constraint satisfaction. S.C. Shapiro (Ed.) Encyclopedia on Artificial Intelligence, John Wiley & Sons, NY, 1987. 68

[180] D. Mangin. *La Ville franchisée : Formes et structures de la ville contemporaine*. Éd. de la Villette, 2004. 27, 31

[181] Marie-Eleonore Marmion. *Recherche locale et optimisation combinatoire : de l'analyse structurelle d'un problème à la conception d'algorithmes efficaces*. These, Université des Sciences et Technologie de Lille - Lille I, December 2011. 79, 90, 223

[182] N. Mars, A. Hornsby, and Dynamic City Foundation. *The Chinese Dream : A Society Under Construction*. 010 Publishers, 2008. 12

[183] J.P. Martin. *Un langage de modélisation à base de règles pour la programmation par contraintes*. PhD thesis, UNIVERSITÉ PIERRE ET MARIE CURIE, 2010. 96

[184] Olivier Martin, Steve W. Otto, and Edward W. Felten. Large-Step Markov Chains for the Traveling Salesman Problem. *Complex Systems*, 5 :299–326, 1991. 83

[185] H. Mathieu and M. Tilmont. Étude sur les relations entre problèmes énergétiques et aménagement urbain. Centre de Recherche d'Urbanisme, 1978. 43

[186] J.L. Maupu. *La ville creuse pour un urbanisme durable*. Editions L'Harmattan, 2006. 36, 38

[187] Thierry Mautor. *Méta-heuristiques et méthodes exactes pour les problèmes d'optimisation combinatoire difficiles : illustration sur le problème d'affectation quadratique*. Hdr, Université de Versailles - Saint-Quentin-en-Yvelines, 2001. 90, 95, 126, 127, 128, 221

[188] P. Merlin. *L'urbanisme*. Que sais-je ? Presses universitaires de France, 2007. 20

[189] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6) :1087–1092, June 1953. 83

[190] Zbigniew Michalewicz, Dipankar Dasgupta, Rodolphe G. Le Riche, and Marc Schoenauer. Evolutionary algorithms for constrained engineering problems. *Evolutionary Computation*, 4 :1–32, 1996. 73

[191] L. Michel and P. Van Hentenryck. Localizer : A Modeling Language for Local Search. In *Third International Conference on the Principles and Practice of Constraint Programming (CP'97)*, Lintz, Austria, October 1997. 96

[192] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing Conflicts : A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 58(1-3) :161–205, 1992. 87, 107, 154

[193] H. Mintzberg. *The structuring of organizations : a synthesis of the research*. Theory of management policy series. Prentice-Hall, 1979. 100, 101

[194] O. Mongin. *Vers la troisième ville ?* Questions de société : savoirs, enjeux, débats. Hachette, 1995. 26, 28

[195] D.C. Montgomery. *Design and Analysis of Experiments*. Number vol. 2. Wiley, 1984. 75

[196] A. Morcheoine, B. Bresse, and J.P. Orfeuil. Énergie, environnement et déplacements urbains : quelques points de repère. Techniques Sciences Méthodes, génie urbain génie rural, n° 1, pages 17-25, 1996. 43

[197] E. Morin. *Introduction à la pensée complexe*. Collection Communication et complexité. ESF, 1990. 114

[198] R.W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Natural Computing Series. Springer, 2004. 60

[199] Sana Moujahed. *Approche multi-agents auto-organisée pour la résolution de contraintes spatiales dans les problèmes de positionnement mono et multi-niveaux*. PhD thesis, Université de Technologie de Belfort-Montbéliard, 2007. 125, 149

[200] Sana Moujahed, Olivier Simonin, and Abderrafiaa Koukam. Location problems optimization by a self-organizing multiagent approach. *Multiagent and Grid Systems*, 5(1) :59–74, 2009. 149

[201] D.J. Musliner, J.A. Hendler, A.K. Agrawala, E.H. Durfee, J.K. Strosnider, and C.J. Paul. The Challenges of Real-Time AI. 28(1) :58–66, 1995. 103

[202] Ioan Negrutiu and Jean-Michel Salles. Les ressources : la capital naturel évanescent et le défi démographique. [88], pages 38–39. 30

[203] Bertrand Neveu, Gilles Trombettoni, and Fred Glover. Id walk : A candidate list strategy with a simple diversification device. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 2004. 75

[204] A. Newell and H.A. Simon. *Human problem solving*. Prentice-Hall, 1972. 101

[205] Yanik Ngoko. *L'Approche du portfolio d'algorithmes pour la construction des algorithmes robustes et adaptatifs*. These, Université de Grenoble, July 2010. 94, 159

[206] T.A.J. Nicholson. *Optimization in Industry : Industrial applications*. London Business School series. Aldine Atherton, 1971. 65

[207] Guy Orcutt. A new type of socio-economic system. *International Journal of Microsimulation*, 1(1) :3–9, 1957. 117

[208] Cindy A. O'Reilly and Andrew S. Cromarty. "Fast" Is Not "Real-Time" : Designing Effective Real-Time AI Systems. *Proc. SPIE*, 0548 :249–257, 1985. 103

[209] Jean-Pierre Orfeuil and Marie-Hélène Massot. La contrainte énergétique doit-elle réguler la ville ou les véhicules ? Mobilités urbaines et réalisme écologique. Annales de la recherche urbaine, n° 103, pages 18-29, 2007. 43

[210] Ibrahim Osman and Gilbert Laporte. Metaheuristics : A bibliography. *Annals of Operations Research*, 63(5) :511–623, October 1996. 73

[211] Ibrahim H. Osman and Nicos Christofides. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, 1(3) :317–336, 1994. 84

[212] I.H. Osman and J.P. Kelly. *Meta-Heuristics : Theory and Applications*. Springer, 1996. 73

[213] A. Oyon. *Une véritable cité ouvrière : Le Familistère de Guise*. Librairie des sciences sociales, 1865. 23

[214] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994. 57

[215] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization : algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982. 60

[216] Julien Perret, Florence Curie, Julien Gaffuri, and Anne Ruas. Un système multi-agent pour la simulation des dynamiques urbaines. In Michel Occello and Lilia Rejeb, editors, *JFSMA*, pages 205–213. Cepadues Editions, 2010. 121

[217] H. Pirenne. *L'origine des constitutions urbaines au moyen âge*. Revue Historique, 1895. 19

[218] Franck Plastria. Continuous Covering Location Problems. [80], pages 37–79. 124, 149

[219] J-Ch Pomerol and S. Barba Romero. *Choix Multicritère dans l'enterprise*. Hermes, Paris, 1993. 101

[220] Jean-Charles Pomerol. Systèmes experts et SIAD : enjeux et conséquences pour les organisations. Technologies de l'Information et Société, Volume 3, n° 1, pages 37-64, 1990. 148

[221] Jean-Charles Pomerol. Artificial intelligence and human decision making. *European Journal of Operational Research*, 99(1) :3 – 25, 1997. 102, 103

[222] Jakob Puchinger and Günther R. Raidl. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization : A Survey and Classification. In *Lecture Notes in Computer Science*, volume 3562, pages 41–53, 2005. 61, 149, 194

[223] D. Pumain, L. Sanders, and T. Saint-Julien. *Villes et auto-organisation*. Economica, 1989. 114

[224] Denise Pumain. Essai sur la distance et l'espace géographique. Revue ATALA - Numéro 12 - La distance, objet géographique, 2009. 113, 114

[225] Ernest G. Ravenstein. The Laws of Migration. *Journal of the Royal Statistical Society*, 52(2) :241–305, June 1889. 113

[226] Jean-Charles Régin. *Modélisation et contraintes globales en programmation par contraintes*. Hdr, Université de Nice Sophia-Antipolis, November 2004. 56

[227] Jean-Charles Régin, Mohamed Rezgui, and Arnaud Malapert. Embarrassingly Parallel Search. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, pages 596–610, 2013. 92, 159

[228] César Rego and Catherine Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In IbrahimH. Osman and JamesP. Kelly, editors, *Meta-Heuristics*, pages 661–675. Springer US, 1996. 94

[229] W.J. Reilly. *The Law of Retail Gravitation*. W.J. Reilly, 1931. 113

[230] Mauricio G. C. Resende. Greedy Randomized Adaptive Search Procedures (GRASP). *Journal of Global Optimization*, 6 :109–133, 1999. 89

[231] Celso C. Ribeiro and Mauricio G. C. Resende. Path-relinking intensification methods for stochastic local search algorithms. *J. Heuristics*, 18(2) :193–214, 2012. 81

[232] P. Riboulet. *Onze Lecons Sur La Composition Urbaine*. Presses de l'École Nationale des Ponts et Chaussées, 1998. 20

[233] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming*. Elsevier Science Inc., New York, NY, USA, 2006. 63, 149, 194

[234] B. Roy. *Méthodologie multicritère d'aide à la décision*. Economica, Paris, 1985. 60, 101

[235] B. Roy. Réflexions sur le thème : Quête de l'optimum et aide à la décision. Cahier du LAMSADE no 167, Université de Paris Dauphine, février 2000. 32 pages. 100

[236] B. Roy and D. Bouyssou. *Aide Multicritère à la Décision : Méthodes et Cas*. Economica, Paris, 1993. 100, 101

[237] S. J. Russell and S. Zilberstein. Anytime sensing, planning, and action : A practical model for robot control. In Ruzena Bajcsy, editor, *Proceedings of the International Conference on Artificial Intelligence (IJCAI-93)*, pages 1402–1407, Chambéry, France, 1993. Morgan Kaufmann publishers Inc. : San Mateo, CA, USA. 99

[238] Sartaj Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3) :555–565, 1976. 125

[239] Nicolas Schabanel. Algorithmes d'approximation et Algorithmes randomisés, 2003. CIRM - ENS LYON. 64, 223

[240] A. Schärlig. *Décider sur plusieurs critères : panorama de l'aide à la décision multicritère*. Collection Diriger l'entreprise. Presses polytechniques romandes, 1985. 60, 101

[241] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued Constraint Satisfaction Problems : Hard and Easy Problems. In *IJCAI-95*, pages 631–639. Morgan Kaufmann, 1995. 67

[242] D.K. Schneider. *Modélisation de la démarche du décideur politique dans la perspective de l'intelligence artificielle*. PhD thesis, 1996. 100

[243] René Séguin, Jean-yves Potvin, Michel Gendreau, Teodor G. Crainic, and Patrice Marcotte. Real-time decision problems : an operations research perspective. *Journal of the Operational Research Society*, 48 :162–174, feb 1997. 103, 104, 106, 178

[244] Marc Sevaux. *Métaheuristiques : Stratégies pour l'optimisation de la production de biens et de services.* Hdr, University of Valenciennes, July 2004. 65

[245] P. Siarry. *Métaheuristiques : Recuits simulé, recherche avec tabous, recherche à voisinages variables, méthodes GRASP, algorithmes évolutionnaires, fourmis artificielles, essaims particulaires et autres méthodes d'optimisation*. Algorithmes. Eyrolles, 2014. 71, 90, 95, 150

[246] SIMAURIF. Rapport 1 : Modèle dynamique de simulation de l'interaction urbanisation-transports en région ile-de-france. Projet de recherche, September 2004. 119, 120, 224

[247] SIMAURIF. Rapport 2 : Modèle dynamique de simulation de l'interaction urbanisation-transports en région ile-de-france. Projet de recherche, October 2005. 120

[248] SIMAURIF. Rapport 3 : Modèle dynamique de simulation de l'interaction urbanisation-transports en région ile-de-france. Projet de recherche, November 2007. 120

[249] Herbert Alexander Simon. *The New Science of Management Decision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1977. 99, 100, 177, 224

[250] Jadranka Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *INFORMS Journal on Computing*, 2(1) :33–45, 1990. 86

[251] Christine Solnon. Cours de programmation par contraintes, 2003. 56

[252] Christine Solnon. *Contribution à la résolution pratique de problèmes combinatoires - des fourmis et des graphes*. Hdr, 2005. 57, 59, 89

[253] Taoufik Souami. De l'écoquartier à la ville intelligente. [88], pages 112–113. 33

[254] R.H. Sprague and E.D. Carlson. *Building effective decision support systems*. Grolier Computer Sciences Library. Prentice-Hall, 1982. 102, 177

[255] Ralph H. Sprague Jr. DSS in context. *Decision Support Systems*, 3(3) :197 – 202, 1987. 102, 177, 224

[256] P. F. Stadler. Towards a Theory of Landscapes. In Lopéz, R. Capovilla, García R. Pelayo, H. Waelbroeck, and F. Zertuche, editors, *Complex Systems and Binary Networks (Proceeding of the Guanajuato Lectures 1995)*, pages 77–163. Springer-Verlag, 1996. 79

[257] P.F. Stadler and W. Schnabl. *The landscape of the traveling salesman problem*. Preprint. Sonderforschungsbereich 343, 1991. 79

[258] John A. Stankovic and K. Ramamritham, editors. *Tutorial : Hard Real-time Systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1989. 99, 150

[259] J. Q. Stewart. Demographic Gravitation : Evidence and Applications. *Sociometry*, 11(1/2) :31–58, 1948. 113

[260] Thomas Stützle. *Local search algorithms for combinatorial problems - analysis, improvements, and new applications.*, volume 220 of *DISKI*. Infix, 1999. 83

[261] Jean-Pierre SUEUR, sénateur. Villes du futur, futur des villes - quel avenir pour les villes du monde ?, June 2011. 19

[262] J. Syrota. *Perspectives énergétiques de la France à l'horizon 2020-2050 : Rapport de la Commission Énergie*. Rapports et documents. Centre d'analyse stratégique - Documentation française, 2008. 43

[263] E. Taillard. Robust tabou search for the quadratic assignment problem. *Parallel Computing*, 17(4-5) :443–455, 1991. 86, 127, 128, 221

[264] Éric D. Taillard, Luca Maria Gambardella, Michel Gendreau, and Jean-Yves Potvin. Adaptive Memory Programming : A Unified View of Meta-Heuristics. *European Journal of Operational Research*, 135(1) :1–16, 2001. 96

[265] E. G. Talbi, Z. Hafidi, and J.-M. Geib. A parallel adaptive tabu search approach. *Parallel Comput.*, 24(14) :2003–2019, December 1998. 94

[266] E.G. Talbi. *Metaheuristics : From Design to Implementation*. Wiley Series on Parallel and Distributed Computing. Wiley, 2009. 58, 60, 61, 64, 65, 71, 73, 78, 79, 91, 94, 95, 159, 223

[267] H. Toussain. *Algorithmique rapide pour les problèmes de tournées et d'ordonnancement*. PhD thesis, Université Blaise Pascal - Clermont-Ferrand II, 2010. 66, 71

[268] Charlotte Truchet, Florian Richoux, and Philippe Codognet. Prediction of parallel speed-ups for las vegas algorithms. In *42nd International Conference on Parallel Processing, ICPP 2013*, pages 160–169. IEEE, 2013. 93

[269] United-Nations. Department of economic and social affairs, population division : World population 2012. 19

[270] United-Nations. Department of economic and social affairs, population division : World urbanization prospects, the 2011 revision, population of urban agglomerations. 19

[271] United-Nations. Department of economic and social affairs, population division : World urbanization prospects, the 2011 revision, urban and rural areas 2011. 19

[272] Pascal Van Hentenryck and Laurent Michel. *Constraint-Based Local Search*. The MIT Press, 2005. 63, 66, 87, 96, 154

[273] Carlos A Vanegas, Daniel G Aliaga, Peter Wonka, Pascal Müller, Paul Waddell, and Benjamin Watson. Modelling the appearance and behaviour of urban spaces. In *Computer Graphics Forum*, volume 29, pages 25–42. Wiley Online Library, 2010. 49

[274] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001. 64

[275] R. Venturi, D.S. Brown, and S. Izenour. *L'Enseignement de Las Vegas ou le symbolisme oublié de la forme architecturale*. Collection architecture. Architecture + recherches. Mardaga, 1987. 28

[276] Gérard Verfaillie and Thomas Schiex. Maintien de solution dans les problèmes dynamiques de satisfaction de contraintes : Bilan de quelques approches, 1995. 106

[277] Bruno Villalba. La dimension politique refoulée du développement durable. [88], pages 26–27. 30

[278] L. von Bertalanffy and J.B. Chabrol. *Théorie générale des systèmes*. Sciences des organisations. Dunod, 1993. 114

[279] J.H. von Thünen and J. Laverrière. *Recherches sur l'influence que le prix des grains, la richesse du sol et les impots exercent sur les systèmes de culture*. 1851. 110, 224

[280] Stefan Voß, Ibrahim H. Osman, and Catherine Roucairol, editors. *Meta-Heuristics : Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Norwell, MA, USA, 1999. 73

[281] Chris Voudouris and Edward Tsang. Guided local search. Technical report, European Journal of Operational Research, 1995. 89

[282] Jens Vygen. Approximation algorithms for facility location problems, 2005. 149

[283] Paul Waddell. UrbanSim : Modeling Urban Development for Land Use, Transportation and Environmental Planning. *Journal of the American Planning Association*, 68(3) :297–314, 2002. 49, 117

[284] D. Walker and T.L. Daniels. *The Planners Guide to CommunityViz : The Essential Tool for a New Generation of Planning*. Orton Family Foundation Books. Planners Press, American Planning Association, 2011. 122

[285] Jacques Weber. Développement viable, durable ou du rabe ? [88], pages 50–51. 30

[286] Marino WIDMER. Les métaheuiristiques : des outils performants pour les problèmes industriels. In *3e Conférence Francophone de MOdélisation et SIMulation : Conception, Analyse et Gestion des Systèmes Industriels*, MOSIM'01, Troyes (France), April 2001. 65

[287] Marc Wiel. Comment construire une ville cohérente ? Études foncières, n° 138, pages 12-17, 2009. 43

[288] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Tech. Rep. No. SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1995. 61

[289] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. pages 355–366, 1932. 78

[290] Mutsunori Yagiura and Toshihide Ibaraki. On metaheuristic algorithms for combinatorial optimization problems. *Systems and Computers in Japan*, 32(3) :33–55, 2001. 95

[291] Shlomo Zilberstein and Stuart Russell. Optimal composition of real-time systems. *ARTIFICIAL INTELLIGENCE*, 82 :181–213, 1996. 99

[292] G. Zipf. Human Behaviour and the Principle of Least-Effort. Addison-Wesley, Cambridge, MA, 1949. 111

[293] G. K. Zipf. The $\frac{P1P2}{D}$ hypothesis : on the intercity movement of persons, 1946. 113

[294] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization : A critical survey. *Annals of Operations Research*, 131 :373–395, 2004. 70

l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Bruno BELIN

Conception interactive d'environnements urbains durables à base de résolution de contraintes

Interactive design of sustainable cities with constraints solving

Résumé
La conception de villes plus durables est devenue un problème de société central. Une ville, à un stade très en amont du processus de conception, peut être vue comme un ensemble équilibré de formes urbaines (unités résidentielles, commerciales, artisanales, industrielles, routes, écoles, parcs, ...) qui doivent être spatialement organisées suivant des règles complexes issues d'une vision systémique de la ville, intégrant les facettes sociales, économiques, écologiques et de transport. Dans cette thèse, nous proposons un outil d'aide à la décision pour assister les urbanistes et les décideurs dans cette tâche de conception de villes durables. Nous modélisons le problème d'aménagement urbain comme un problème d'optimisation que nous résolvons par des techniques de recherche locale en réalisant une succession de permutations entre formes urbaines. Notre outil organise alors automatiquement les formes urbaines sur un territoire vierge, et délimité dans l'espace. Nous étendons cet algorithme séquentiel avec des heuristiques novatrices pour améliorer les temps de calcul, et nous proposons une version distribuée efficace pour des problèmes de grande taille. Enfin, nous y ajoutons des fonctionnalités interactives permettant aux experts de modifier l'organisation spatiale de la ville, tout en maintenant à la volée les relations entre les formes urbaines, et en les informant des impacts de leurs choix. Les avantages de notre approche sont mis en évidence par des exemples et des évaluations d'experts du domaine, et ouvrent la voie vers de nouvelles façons de concevoir les villes du futur.

Mots clés
Système interactif d'aide à la décision, recherche locale distribuée, planification urbaine, ville durable.

Abstract
The design of more sustainable cities has emerged as a central society issue. A city, in the early stage of its design process, can be seen as a balanced set of urban shapes (residential, commercial, artisanal and industrial units, roads, schools, parks, ...). These shapes need to be spatially organized following complex rules based on a systemic view of the city, including social, economic, ecological and transportation aspects. In this thesis, we provide a computer-aided decision tool to assist urban planners and decision makers in the task of designing sustainable cities. We cast the urban planning problem as an optimization problem that we solve with local search techniques, by iteratively swapping the urban shapes. At this stage, our tool automatically organizes urban shapes over a given empty, spatially delimited territory. We extend this sequential algorithm with novel heuristics to improve the computation time, and propose a distributed version, efficient on large problems. Finally, we add interactive features that allow the experts to modify the spatial organization of the city, while maintaining on the fly the relations between shapes and informing the experts of the impacts of their choices. The benefits of our approach are highlighted by examples and feedbacks from experts in the domain, and open the way for new approaches to design the cities of the future.

Key Words
Decision support system, distributed local search, urban planning, sustainable city.